SAMS PRESENTS

22247

TI-99/4A: 24 BASIC Programs

by Carol Ann Casciato
and Don Horsfall

TI-99/4A: 24 BASIC Programs

Carol Ann Casciato and **Donald J. Horsfall** are the principals in International Technical Communications, Inc., a computer systems research and consulting firm in the Philadelphia area. For the last 12 years, they have done management and systems consulting, research, and writing for a variety of Fortune 500 clients.

Their first exposure to professional writing came when they produced more than 25 manuals for a large technical documentation project. Since that time, they have written in-depth computer industry research reports, detailed technical product analyses, and manuals for microcomputer manufacturers.

When not reading, writing, or consulting on computers, Carol Ann prepares elaborate chocolate desserts and cares for her large collection of exotic plants. Don's interests include science fiction, restoring his Victorian home, and collecting space art.

TI-99/4A: 24 BASIC Programs

by

Carol Ann Casciato and

Donald J. Horsfall

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
SECOND PRINTING—1984

International Standard Book Number: 0-672-22247-7
Library of Congress Catalog Card Number: 83-50831

Edited by: *Jim Rounds*

Printed in the United States of America.

Preface

The programs in this book were written to serve two essential purposes. First, of course, to provide useful and/or fun programs for you to run on your TI-99/4A. The twenty-four programs in this book run the gamut from the most frivolous, if venerable, entertainment programs for children and adults, to utilities to assist all members of the family around the house, to service programs for computer buffs.

The second purpose is to provide a starting point for learning personal programming on your home computer. Throughout this book, we encourage you to experiment with programming. It's very difficult to begin, as a novice programmer, with just an idea and turn that idea into a fully functioning program. It is much easier to begin with something already running and then fiddle with it until it does *exactly* what you want it to do.

Most of these programs were tested on a group of cheerfully suffering friends and relatives who happen to own TI-99/4A computers. Our most rewarding feedback comes not when they detect errors in our otherwise impeccable code, but when they see a way to modify a program so that it meets their particular needs.

As an example, one of our testers, who travels quite a bit, is adapting the Trip Planning program to create a Trip History program that will keep track of travel expenses over a long trip. This is the essence of personal programming on your home computer.

CAROL ANN CASCIATO AND DONALD J. HORSFALL

ACKNOWLEDGMENTS

For suffering through a seemingly endless string of "Here, try this program!", we thank our TI-99/4A owning friends and relatives: Abby, Anna Marie, Betty, Betty Ann, Eleanor, Joe, Matthew, Roger, and Terry. And thanks, too, to our music director, Robert W. Cole.

Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

1

Introduction

The programs in this book are designed to make it easy for you to enter them, get them running, and make your own modifications.

All the programs are written in TI BASIC, requiring only the console and a cassette tape (if you want to save them).

In this chapter, we're going to look at the structure of the programs, how you can best enter them, and how to get the programs running successfully.

For those of you who are interested in programming, one of the best ways to improve your programming, skills and increase your knowledge is to make customizing changes in existing programs. We encourage you to add features, bend processes to your own needs, even remove features you don't want.

To help you in your efforts to make these programs do exactly what you want, we're going to give you some ideas on how to go about modifying the programs in the easiest way.

Statement Numbers

Many of these programs are large, over 250 statements; the Space War game contains 495 statements. This is quite large by home computer program book standards and is a direct result of having 16 Kilobytes of memory to use.

Most BASIC programs you see are sequentially line numbered, beginning with 10 or 100 and proceeding by tens to the end. This is suitable for smaller programs, but it makes it difficult to follow the logic in a larger program, and very difficult to modify or expand the program—especially if you don't have a printer. A few minor addi-

tions, some resequencing, and soon the listing in the book bears no relationship to the program you're running.

We've attempted to avoid this problem, and to provide support for your understanding of the logic of these programs, by grouping statements that perform a logical function into their own statement number range. This provides space for you to add your own lines to the programs without making the listing in the book totally obsolete, and makes it easier for you to see where and how individual logical functions in the program are carried out.

Entering Programs: the NUM Command

The TI-99/4A is the easiest home computer on which to enter programs. Its **NUM** command automatically generates statement numbers, saving you the considerable hassle of keying in these line numbers yourself.

The NUM command has the format:

NUM starting-line-number,increment

Where:

starting-line-number is the first line number you want to enter

increment is the amount added to each line number to generate the next line number.

If you don't supply a starting line number, the TI-99/4A begins with line 100. If you don't supply an increment, it increments by 10.

Although starting line numbers vary, all of the programs in this book increment by 10.

Let's assume you're going to enter the Present Value program. In the following sample dialog:

- your commands are in *italics*
- the TI-99/4A's responses in **bold face**
- >ENTER< means you press the ENTER key.

If you're unfamiliar with the NUM command, try following this example on your TI computer. You don't have to put all the statements in if you aren't interested in entering the Present Value program; just end a section wherever you feel like it. You begin by entering.

<NUM<ENTER>
100 ***REM PRESENT VALUE PROGRAM***<ENTER>

```
110 DEF RD(X)=INT (X*100+.5)/100<ENTER>
120 CALL CLEAR<ENTER>
    .
    .
    .
```

The TI computer continues to put statement numbers on the screen until you enter an empty line (just the ENTER key with no other letters on the line). In the Present Value program, it would look like this:

```
    .
    .
    .
240 GOTO 160<ENTER>
250 STOP<ENTER>
260 <ENTER>
>
```

At this point, you've finished the first logical section of code in the Present Value program. *To protect yourself from unfortunate errors,* resulting in loss of your program, you should make a habit of saving the program to tape, Wafertape, or diskette at regular intervals. The end of a logical program section is a good place to stop for a SAVE.

You'll notice in the Present Value program listing that the next line number is 1000. You proceed as follows:

```
>NUM 1000<ENTER>
1000 REM PV OF A LUMP SUM<ENTER>
1010 CALL CLEAR<ENTER>
1020 PRINT "Enter the desired"<ENTER>
1030 INPUT " lump sum payment->":LSUM<ENTER>
    .
    .
    .
```

As before, the computer continues to put line numbers on the screen until you tell it to stop by entering an empty line. The last line number in the 1000 series of the Present Value program is 1190, so you would stop like this:

```
    .
    .
    .
1190 RETURN<ENTER>
1200 <ENTER>
>
```

The 2000, 3000, and 4000 series are entered in much the same way.

Saving the Program: the SAVE Command

Saving a program for recall later is as easy as:

SAVE *device-name.file-name*

Where:

device-name is the name of a storage device like CS1 for a cassette, or DSK1 for disk drive 1.
file-name is the name of the file on the device. You don't use a *file-name* when saving to a cassette tape.

Of course, you want to save your program after you've finished entering it. But you should also SAVE it from time to time as you enter it. A good time to save an insurance copy (that insures you won't lose the whole thing in the event of a problem) is at the end of a logical section of the program. This isn't an iron-clad rule; anytime you feel you've invested so much time and effort to enter the program that you'd really hate to lose it, SAVE it.

If you're saving to a cassette tape, simply enter:

SAVE CS1

When you're entering a program, you can place a tape in your cassette recorder and SAVE to it repeatedly. There's no need to rewind after every SAVE, just write down the tape counter number each time so that you can readily find the program if you should have to recover it.

Loading a Program: the OLD Command

Before a program can be run, it must be loaded into the computer's memory. The program may reside on cassette tape, Wafertape, or on a diskette, depending on which devices you have attached to your TI computer. In all cases, you use the OLD command to move the program from the device to Random Access Memory where it can be executed.

The format of the OLD command is:

OLD *device-name.file-name*

Where:

device-name is the name of a storage device like CS1 for a cassette, or DSK1 for disk drive 1.
file-name is the name of the file on the device. You don't use a *file-name* when loading a program from a cassette tape.

To load a program from a cassette tape, you enter the command:

OLD CS1

If you have a dual cassette cable, note that you can't load from device CS2—CS2 is a write-only device.

Getting It Running: Debugging Entered Programs

You've typed in the program, saved it so you won't accidentally lose it, and entered the RUN command . . . And—BOOP—it FAILS!

Some horrible error message comes up on your screen as the computer waits for your next command.

What should you do? Primarily, DON'T PANIC! It's only a machine and can be made to do what you want.

The first thing to do is *look at the statement where the error occurred* (if the message lists one). Compare it to the statement in the listing in the book. If you see any obvious problems, fix them and RUN it again.

If you don't see a problem with the statement, *check the program to make sure you entered all of the statements.* If the program is a large one, divided into sections, you can look at the beginning and end of each section to be sure the line numbers match those in the listing. Sample a few lines in the middle too, just to see that they're in the right place with respect to the listing.

If you find that a section is too short, look for the missing statement(s) and insert it/them where they belong. You may have to fix all the statements following any that are missing, up to the end of the section, because GOTO, IF–THEN–ELSE, and GOSUB statements may refer to line numbers now on the wrong statements.

If it looks as though you've got all the code entered in the right place, chances are you've got a *misspelled variable name* somewhere. This can be very tedious to find.

You can use the brute force approach of looking through the whole program, comparing it line for line with the listing in the book. And you may, in the end, have to do just that. But first, at least, see if you can figure out what is wrong in the statement that failed.

You can PRINT the value of the variables referred to in the statement. For example, if the statement that failed reads:

```
1020 CALL COLOR(I,FGCOLOR,BGCOLOR)
```

You can find out the value of the variables involved by entering:

```
PRINT I,FGCOLOR,BGCOLOR
```

Once you see the value of these variables, and compare their values to those allowed, or reasonable, in the statement, you may be able to tell which variable is bad.

Having determined which variable has a bad value, you can restrict your search through the program listing in the book to those statements which refer to that variable. If you find it calculated somewhere, PRINT the values of the variables involved in calculating it, just as you did for those in the statement that failed. If you keep tracing it back, you'll eventually find the error.

Actually, this is not as lengthy a process as it sounds. You will find most of these kinds of errors after a few minute's look at the program.

Another cause of program failure, often less easy to find, is a logic error introduced into the code by a mistyped line or by typing the wrong line number in a GOTO, IF-THEN-ELSE, or GOSUB statement.

This kind of problem can show up as a bad value somewhere else in the program, or as a wrong answer. As you trace through the variables whose values appear to be wrong, pay attention to the IF statements, GOTO statements, and GOSUB statements that affect how those variables are calculated. You might find a statement that reads:

```
3050 IF K>0 THEN 3080
```

When it should, in fact, read:

```
3050 IF K<>0 THEN 3080
```

When all else fails, go through the program line by line, comparing it to the listing in the book. This is often much easier if two people work on it together. One should read, aloud, the program as it appears on the screen while the other compares that to what is in the book.

Custom Programming: Modification Made Easy

The advantage of having your own computer is being able to have it your own way. Some modifications you can make to the programs may be small, but they contribute to the ease with which you use them on your TI-99/4A.

For example, if you enter a program that saves data to a file, you can modify the file writing and retrieval sections to access only the device you have. That is, if you have only a cassette tape, why ask for a file name each time? Skip that question and go directly to CS1. Likewise, if you have one disk drive, change the program so that it automatically adds "DSK1." to the file name you enter.

Making small changes like this adds to the pleasure of using your TI-99/4A and provides you the opportunity to learn some programming while accomplishing some useful goal (unlike those exercises in program learning books that seem to have no purpose).

In the sections devoted to each program, you find two tables:

- a table of the statement number ranges of routines that perform some task in the program, and
- a table containing the names and descriptions of the key variables in the program.

You will notice as you look at these programs that they have a *modular structure*—functions are performed in their own sections of code, separate from other functions in the program.

In some cases, your change may involve inserting a few lines between existing lines. Or, you may want to add options to the end of an existing sequence of code; as in adding a new conversion to the conversion routine, or a different stitch to the needlepoint program.

In both of these situations, you can simply insert or add the code—we've left room for these kinds of changes in the line number sequences. You'll also, of course, have to adjust the menus or whatever to make use of the new code.

If you are going to make changes or additions to the program that involve more than inserting a few lines between existing lines, we encourage you to add those lines in their own statement number group at the end of the program (or at some other convenient spot). You then access your new code using a GOSUB statement at the point where you would have inserted it in-line.

Look at the way the period of an annuity is calculated in the Present Value program (statement ranges 3000 and 4000) to see how convenient it is to use single function routines accessed via a GOSUB statement. As you can see, the 3000 series routine is accessed from an ON-GOSUB statement at line 230. The 3000 series routine handles the interaction with the user and then GOSUBs to the 4000 series routine to actually perform the search for the period of annuity. On return from the 4000 series routine, the 3000 series routine prints the results to the screen.

Using Cassette Tape

If you plan to use a program that saves data on a file and you are using cassettes for file storage, look at the OPEN statements in the program.

If the OPEN statement lacks a FIXED attribute, add it. For example, if the OPEN statement reads:

```
OPEN #20:FILE$, OUTPUT, INTERNAL
```

change it for use with a cassette tape to:

```
OPEN #20:FILE$, OUTPUT, INTERNAL, FIXED 192
```

The 192 is the fixed length block size on the tape. You may not need a block this large, but if you don't know what block size you need, use 192.

2

Games

This chapter gives you instructions and code for these great games:

- BAGELS: Guess the three-digit number.
- BLACKJACK: See how much you can win.
- EXTRATERRESTRIAL TYPING: An exotic typing program to help you learn the keyboard. Capture those letters before they escape!
- PLANET FALL: Try to land on several planets without crashing your ship.
- SPACE WAR 7: Save the Federation from the treacherous Krinng.
- TIC-TAC-TOE: Play against the computer or another person.
- WUMPUS: Hunt the Wumpus. But beware the HAZARDS!

BAGELS PROGRAM

What it does:

Bagels is a fun game for children and adults. One of the classic computer games, Bagels teaches logic skills as it entertains.

The object of the game is to guess a three-digit number the computer has picked at random in as few guesses as possible. Each time you enter your three-digit guess, the computer gives hints to tell you how close to the number you are. Don't forget that *zero* is a valid digit.

If you guess a correct digit, but in the wrong place, the program prints **PICO.** For example, suppose the computer chose the number 123 and you guess 256. The program prints PICO to tell you that you got one digit right (the 2), but that it is in the wrong place (first instead of second).

If you guess a correct digit in the correct place, the program prints

FERMI. Again assume the computer picks the number 123. If you guess 453 the computer prints the hint FERMI, indicating that you've got one digit right (the 3) and it's in the right place.

If your guess has no correct digits, the program prints **BAGELS.**

What it shows you:

Bagels is a very simple program that performs an easily understood function. If you're just learning to program in BASIC, look through this program to see how various commonly encountered programming problems are solved.

Note, for example, the methods used to choose randomly a three-digit number (statements 320–360) in which no two digits are the same, or the processing of your answer in statements 410–480.

What you can do:

You can easily make this game harder by increasing the size of the number from three to four or five digits. Note that it might take more than 20 guesses to find the correct four- or five-digit number.

Table 2-1. Bagels Routines

Lines	Function
100-300	Initialization and game instructions.
310-360	Select a random three-digit number.
370-480	Get and error check a guess.
490-780	Compare the guess to the answer and print the hints.
790-910	Error messages and end-of-game.

Table 2-2. Bagels Variables

Variable	Description
ANSWER(3)	The three digits of your answer.
FERMIS	The number of correct digits in the right position.
MATCH(3)	The three digits of the randomly selected correct answer.
PICOS	The number of correct digits in the wrong place.

Listing 2-1. Bagels Program

```
100 CALL CLEAR
110 PRINT TAB(3);"******* BAGELS *******"
120 PRINT : : : : : : : : : :
130 FOR I=1 TO 200
140 NEXT I
150 CALL CLEAR
160 DIM MATCH(3),ANSWER(3)
170 NUMBERS$="0123456789"
```

Listing 2-1—cont. Bagels Program

```
180 RANDOMIZE
190 INPUT "Want instructions? ":Y$
200 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 310
210 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 240
220 PRINT "Answer yes or no, please."
230 GOTO 190
240 CALL CLEAR
250 PRINT :"I am thinking of a three":"digit number."
260 PRINT :"You guess what number":"I have in mind."
270 PRINT :"I will give you hints:"
280 PRINT :"PICO   - One digit is in";TAB(11);
    "the wrong place"
290 PRINT :"FERMI  - One digit is in";TAB(11);
    "the right place"
300 PRINT :"BAGELS - No digit is correct"
310 MATCH(1)=INT(10*RND)
320 MATCH(2)=INT(10*RND)
330 IF MATCH(2)=MATCH(1)THEN 320
340 MATCH(3)=INT(10*RND)
350 IF (MATCH(3)=MATCH(1))+(MATCH(3)=MATCH(2))THEN 340
360 PRINT :"Ok.  I have a number."
370 FOR I=1 TO 20
380 PRINT :"Guess #";I;":";
390 INPUT A$
400 IF LEN(A$)<>3 THEN 810
410 FOR J=1 TO 3
420 ANSWER(J)=POS(NUMBERS$,SEG$(A$,J,1),1)-1
430 IF ANSWER(J)<>-1 THEN 470
440 CALL SOUND(300,131,0,262,2,-2,1)
450 PRINT : :"WHAT????"
460 GOTO 380
470 NEXT J
480 PRINT ANSWER(1);ANSWER(2);ANSWER(3);
490 IF (ANSWER(1)=ANSWER(2))+(ANSWER(2)=ANSWER(3))+
    (ANSWER(3)=ANSWER(1))THEN 840
500 PICOS=0
510 FERMIS=0
520 FOR J=1 TO 2
530 IF MATCH(J)<>ANSWER(J+1)THEN 550
540 PICOS=PICOS+1
550 IF MATCH(J+1)<>ANSWER(J)THEN 570
560 PICOS=PICOS+1
570 NEXT J
580 IF MATCH(1)<>ANSWER(3)THEN 600
590 PICOS=PICOS+1
600 IF MATCH(3)<>ANSWER(1)THEN 620
610 PICOS=PICOS+1
620 FOR J=1 TO 3
630 IF MATCH(J)<>ANSWER(J)THEN 650
640 FERMIS=FERMIS+1
650 NEXT J
660 IF FERMIS=3 THEN 870
670 PRINT TAB(10);
680 IF PICOS=0 THEN 720
690 FOR J=1 TO PICOS
700 PRINT "PICO ";
710 NEXT J
720 IF FERMIS=0 THEN 760
```

Listing 2-1—cont. Bagels Program

```
730 FOR J=1 TO FERMIS
740 PRINT "FERMI ";
750 NEXT J
760 IF PICOS+FERMIS THEN 780
770 PRINT "BAGELS";
780 NEXT I
790 PRINT "Oh well!"
800 GOTO 880
810 CALL SOUND(300,131,0,262,2,-2,1)
820 PRINT :"Guess a three digit number!!"
830 GOTO 380
840 PRINT : :"Oh. I forgot to tell you":
    "that the number I have"
850 PRINT "in mind has no two digits":"the same."
860 GOTO 380
870 PRINT : : :TAB(10);"YOU GOT IT!"
880 PRINT : : : : : :
890 INPUT "Play again? ":Y$
900 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 180
910 END
```

BLACKJACK PROGRAM

What it does:

Blackjack is one of the most popular games in the world's casinos because it's a fast moving game that's easy to understand. The object is to draw cards, adding up their values, until you are as close to 21 as possible without going over.

The numbered cards are worth the points they show. The face cards are worth 10 points. The aces are worth either 1 point or 11 points, whichever you need.

In the first deal, you get one card down and one up. If the total of these two cards is 21, you have BLACKJACK and win immediately. Otherwise, you can either draw another card or go with what you have.

The dealer (computer) in this game will always draw a card on a total less than 17.

You have several options at some points in the game. On the first two cards, you can choose to double your bet.

If the dealer is showing an ace, you can play an insurance bet. In this case, you break even if the dealer has 21.

If you are holding a pair, you may split the hand into two hands, each with the original bet. If the split pair are aces, you are allowed only one more card in each hand.

Once you've placed your bet, this Blackjack program recognizes the following commands:

- **HIT** (or H) tells the program to deal you another card.

- **GOOD** (or G) tells the computer that you don't want any more cards and will stand with what you have.
- **DOUBLE** (or D) doubles your existing bet.
- **SPLIT** (or S) splits your hand into two if you have a pair.
- **END** (or E) ends the game.

What it shows you:

This is a fairly easy game program. If you're interested in programming, you might look at how commands are decoded using the POS function and an ON-GOTO in statements 1400–1480. Notice that a lower-case letter can be converted to upper case by subtracting 32 from its ASCII value.

What you can do:

Cards are written to the screen in 11000 series routine. You can convert this to a graphic output routine in a variety of ways. You could simply use the Character Editor to design Heart, Diamond, Club, and Spade characters to display in place of the words.

Or, you can get very ambitious and actually draw the cards on the screen. If you want to try this, look at the Tic-Tac-Toe program, the

Table 2-3. Blackjack Routines

Lines	Function
100-260	Initialization dialog.
1000-1500	Main game control code.
2000-2320	Player hit routine (player draws a card).
3000-3050	Player stands with current hand.
4000-4200	Doubling routine.
5000-5430	Hand split routine.
6000-6040	Construct the deck to deal from.
7000-7110	Shuffle the deck.
8000-8220	Deal the first two cards to the dealer and player.
9000-9100	Compute the value of the dealer's hand.
10000-10090	Compute the value of a card.
11000-11310	Print a card to the screen.
12000-12100	Compute the value of the player's hand.
13000-13130	Shuffle the remainder of the deck.
14000-14470	Dealer's play logic.
15000-15090	Print the player's win/loss standing.
16000-16110	Place the insurance bet.
17000-17110	End of game wrap-up.
18000-18280	Print game rules.
19000-19040	Delay routine.

Table 2-4. Blackjack Variables

Variable	Description
ACE(16)	Counts the number of aces counting as 11 in the player's hand(s).
BET	The amount of the player's bet.
BETS(16)	The player's bets in the hands played.
CARD	The current card used in the evaluate and print card routines.
CCNT(16)	Count of the cards played in each hand.
DACES	Number of aces in dealer's hand.
DECK(52)	The deck cards are dealt from.
DLR(11)	The dealer's cards.
DVAL	The value of the dealer's hand.
F$	Player's command.
HAND	The hand currently being played.
LAST	Subscript of the last card in the DECK array.
MONEY	The money the player has won (lost).
NXT	Subscript of the next card to be dealt from the DECK array.
PLYR(16,11)	The player's cards for each hand played.
PVAL(16)	The value of the player's hand in each HAND played.

Exotic Typing program, and the Planet Fall program to see how advanced graphics work is carried out in TI BASIC.

Listing 2-2. Blackjack Program

```
100 REM BLACKJACK
110 DIM DECK(52),BETS(16),ACE(16),PLYR(16,11),DLR(11),
    CCNT(16),PVAL(16)
120 RANDOMIZE
130 CALL CLEAR
140 CALL SCREEN(13)
150 PRINT :TAB(4);"WELCOME TO CASINO 99": :TAB(8);
    "BLACKJACK": : : : :
160 PRINT : :"Do you want"
170 INPUT "   instructions(Y/N)->":Y$
180 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 210
190 IF (SEG$(Y$,1,1)<>"Y")*(SEG$(Y$,1,1)<>"y")THEN 160
200 GOSUB 18000
210 ROUND=16
220 LAST=52
230 GOSUB 6000
240 BGN=1
250 GOSUB 7000
260 NXT=1
1000 PRINT : :
1010 HAND=1
1020 INPUT "Your bet-> ":BET
1030 IF BET>0 THEN 1070
1040 BGN=1
1050 GOSUB 7000
1060 GOTO 1010
```

Listing 2-2—cont. Blackjack Program

```
1070 IF BET<=5000 THEN 1100
1080 PRINT : :"Sorry.  The house limit is $5000!": :
1090 GOTO 1020
1100 GOSUB 8000
1110 BETS(1)=BET
1120 CARD=DLR(2)
1130 PRINT :"My up card";
1140 GOSUB 11000
1150 CARD=PLYR(ROUND,1)
1160 PRINT :"Your 1st card";
1170 GOSUB 11000
1180 PRINT "Your 2nd card";
1190 CARD=PLYR(ROUND,2)
1200 GOSUB 11000
1210 GOSUB 12000
1220 IF TMP<>11 THEN 1240
1230 GOSUB 16000
1240 IF DVAL<>21 THEN 1330
1250 PRINT :"I have BLACKJACK, ";
1260 IF PVAL(1)<>21 THEN 1300
1270 PRINT "So do you, we push."
1280 GOSUB 15000
1290 GOTO 1000
1300 PRINT : : :TAB(10);"YOU LOSE!!!!!": : : : :
1310 MONEY=MONEY-BET
1320 GOTO 1280
1330 IF PVAL(1)<>21 THEN 1370
1340 PRINT : : :
     "You have BLACKJACK.  You WIN!!!!!": : : : :
1350 MONEY=MONEY+1.5*BET
1360 GOTO 1280
1370 PRINT :"Play ";
1380 IF ROUND=1 THEN 1400
1390 PRINT "for hand";HAND;
1400 PRINT :"Your total is";PVAL(HAND): :
1410 INPUT "Your play->":F$
1420 IF LEN(F$)=0 THEN 1410
1430 F$=SEG$(F$,1,1)
1440 IF ASC(F$)<90 THEN 1460
1450 F$=CHR$(ASC(F$)-32)
1460 C=POS("HGDSE",F$,1)
1470 IF C<>0 THEN 1500
1480 PRINT : :"Commands are H G D S E.": :
1490 GOTO 1410
1500 ON C GOTO 2000,3000,4000,5000,17000
2000 REM PLAYER HIT ROUTINE
2010 IF NXT<=LAST THEN 2030
2020 GOSUB 13000
2030 TMP=CCNT(HAND)
2040 TMP=TMP+1
2050 CCNT(HAND)=TMP
2060 CARD=DECK(NXT)
2070 PLYR(HAND,TMP)=CARD
2080 PRINT "Your card is";
2090 GOSUB 11000
2100 GOSUB 10000
2110 NXT=NXT+1
2120 IF CARD<>11 THEN 2140
```

Listing 2-2—cont. Blackjack Program

```
2130 ACE(HAND)=ACE(HAND)+1
2140 PVAL(HAND)=PVAL(HAND)+CARD
2150 IF PVAL(HAND)<22 THEN 1370
2160 IF ACE(HAND)=0 THEN 2200
2170 ACE(HAND)=ACE(HAND)-1
2180 PVAL(HAND)=PVAL(HAND)-10
2190 GOTO 2150
2200 PRINT :"You busted with";PVAL(HAND)
2210 PVAL(HAND)=0
2220 Y=Y-1
2230 PRINT
2240 REM CHECK FOR END OF PLAY
2250 IF HAND<ROUND THEN 2280
2260 GOSUB 14000
2270 GOTO 1000
2280 HAND=HAND+1
2290 CARD=PLYR(HAND,1)
2300 PRINT "Your 1st card for hand ";HAND;" was";
2310 GOSUB 11000
2320 GOTO 1370
3000 REM PLAYER STAND ROUTINE
3010 IF PVAL(HAND)<22 THEN 2250
3020 IF ACE(HAND)=0 THEN 2200
3030 PVAL(HAND)=PVAL(HAND)-10
3040 ACE(HAND)=ACE(HAND)-1
3050 HAND=HAND+1
4000 REM DOUBLE DOWN ROUTINE
4010 IF CCNT(HAND)=2 THEN 4040
4020 PRINT : :"Double on 1st 2 cards only."
4030 GOTO 1370
4040 IF NXT<=LAST THEN 4060
4050 GOSUB 13000
4060 BETS(HAND)=2*BET
4070 CARD=DECK(NXT)
4080 PLYR(HAND,3)=CARD
4090 NXT=NXT+1
4100 PRINT "You draw the";
4110 GOSUB 11000
4120 GOSUB 10000
4130 IF CARD<>11 THEN 4150
4140 ACE(HAND)=ACE(HAND)+1
4150 PVAL(HAND)=PVAL(HAND)+CARD
4160 IF PVAL(HAND)<22 THEN 2250
4170 IF ACE(HAND)=0 THEN 2200
4180 ACE(HAND)=ACE(HAND)-1
4190 PVAL(HAND)=PVAL(HAND)-10
4200 GOTO 14230
5000 REM SPLIT HAND
5010 CARD=PLYR(HAND,1)
5020 Y=Y+1
5030 GOSUB 10000
5040 TMP=CARD
5050 CARD=PLYR(HAND,2)
5060 GOSUB 10000
5070 IF TMP=CARD THEN 5110
5080 PRINT : :"You may only split pairs."
5090 GOTO 1370
5100 REM PAIR SPLIT ROUTINE
```

Listing 2-2—cont. Blackjack Program

```
5110 ROUND=ROUND+1
5120 Y=Y+1
5130 PLYR(ROUND,1)=PLYR(HAND,2)
5140 CCNT(HAND)=1
5150 CCNT(ROUND)=1
5160 PVAL(HAND)=PVAL(HAND)/2
5170 PVAL(ROUND)=PVAL(HAND)
5180 BETS(ROUND)=BET
5190 IF CARD<>11 THEN 1370
5200 REM ACES WERE SPLIT - 1 CARD EACH
5210 IF NXT<=LAST THEN 5230
5220 GOSUB 13000
5230 CARD=DECK(NXT)
5240 PLYR(HAND,2)=CARD
5250 PRINT "1st ACE gets a";
5260 GOSUB 11000
5270 GOSUB 10000
5280 IF CARD<>11 THEN 5300
5290 CARD=1
5300 PVAL(HAND)=PVAL(HAND)+CARD
5310 NXT=NXT+1
5320 IF NXT<=LAST THEN 5340
5330 GOSUB 13000
5340 CARD=DECK(NXT)
5350 PLYR(ROUND,2)=CARD
5360 PRINT "2nd ACE gets a";
5370 GOSUB 11000
5380 GOSUB 10000
5390 IF CARD<>11 THEN 5410
5400 CARD=1
5410 PVAL(ROUND)=PVAL(ROUND)+CARD
5420 NXT=NXT+1
5430 GOTO 2260
6000 REM BUILD DECK
6010 FOR K=1 TO 52
6020 DECK(K)=K
6030 NEXT K
6040 RETURN
7000 REM SHUFFLE THE CARDS
7010 CALL CLEAR
7020 PRINT :"I'm shuffling.... "
7030 FOR I=BGN TO LAST
7040 C=RND*LAST
7050 IF C<BGN THEN 7040
7060 L=DECK(I)
7070 DECK(I)=DECK(C)
7080 DECK(C)=L
7090 NEXT I
7100 NXT=BGN
7110 RETURN
8000 REM DEAL THE CARDS
8010 FOR I=1 TO 11
8020 DLR(I)=0
8030 FOR J=1 TO ROUND
8040 PLYR(J,I)=0
8050 NEXT J
8060 NEXT I
8070 ROUND=1
```

Listing 2-2—cont. Blackjack Program

```
8080 Y=1
8090 IF NXT+4<=LAST THEN 8120
8100 BGN=1
8110 GOSUB 7000
8120 PRINT :TAB(10);"DEALING..."
8130 PLYR(ROUND,1)=DECK(NXT)
8140 DLR(1)=DECK(NXT+1)
8150 PLYR(ROUND,2)=DECK(NXT+2)
8160 DLR(2)=DECK(NXT+3)
8170 NXT=NXT+4
8180 T=2
8190 CCNT(1)=2
8200 GOSUB 9000
8210 TMP=CARD
8220 RETURN
9000 REM COMPUTE VALUE OF DEALERS HAND
9010 DACES=0
9020 DVAL=0
9030 FOR I=1 TO 2
9040 CARD=DLR(I)
9050 GOSUB 10000
9060 IF CARD<>11 THEN 9080
9070 DACES=DACES+1
9080 DVAL=DVAL+CARD
9090 NEXT I
9100 RETURN
10000 REM COMPUTE THE VALUE OF A CARD
10010 IF CARD<14 THEN 10040
10020 CARD=CARD-13
10030 GOTO 10000
10040 IF CARD<>1 THEN 10070
10050 CARD=11
10060 RETURN
10070 IF CARD<11 THEN 10060
10080 CARD=10
10090 RETURN
11000 REM PRINT A CARD
11010 I=0
11020 IF CARD<14 THEN 11060
11030 CARD=CARD-13
11040 I=I+1
11050 GOTO 11020
11060 IF CARD<>1 THEN 11090
11070 PRINT TAB(17);"ACE ";
11080 GOTO 11220
11090 IF CARD>=10 THEN 11120
11100 PRINT TAB(18);CARD;
11110 GOTO 11220
11120 IF CARD<>10 THEN 11150
11130 PRINT TAB(17);CARD;
11140 GOTO 11220
11150 IF CARD<>11 THEN 11180
11160 PRINT TAB(16);"JACK ";
11170 GOTO 11220
11180 IF CARD<>12 THEN 11210
11190 PRINT TAB(15);"QUEEN ";
11200 GOTO 11220
11210 PRINT TAB(16);"KING ";
```

Listing 2-2—cont. Blackjack Program

```
11220 PRINT "OF ";
11230 ON I+1 GOTO 11240,11260,11280,11300
11240 PRINT "SPADES"
11250 RETURN
11260 PRINT "HEARTS"
11270 RETURN
11280 PRINT "DIAMONDS"
11290 RETURN
11300 PRINT "CLUBS"
11310 RETURN
12000 REM COMPUTE VALUE OF PLAYERS HAND
12010 ACE(HAND)=0
12020 PVAL(HAND)=0
12030 FOR I=1 TO 2
12040 CARD=PLYR(HAND,I)
12050 GOSUB 10000
12060 PVAL(HAND)=PVAL(HAND)+CARD
12070 IF CARD<>11 THEN 12090
12080 ACE(HAND)=ACE(HAND)+1
12090 NEXT I
12100 RETURN
13000 REM RE-SHUFFLE ROUTINE
13010 K=T
13020 FOR I=1 TO ROUND
13030 K=K+CCNT(I)
13040 NEXT I
13050 FOR I=1 TO K
13060 NXT=NXT-1
13070 J=DECK(I)
13080 DECK(I)=DECK(NXT)
13090 DECK(NXT)=J
13100 NEXT I
13110 BGN=K+1
13120 GOSUB 7000
13130 RETURN
14000 REM DEALERS LOGIC
14010 CARD=DLR(1)
14020 PRINT "Dealer's hole card";
14030 GOSUB 11000
14040 IF Y=0 THEN 14290
14050 IF DVAL<17 THEN 14100
14060 IF DVAL>17 THEN 14220
14070 IF DACES=0 THEN 14280
14080 DVAL=DVAL-10
14090 DACES=DACES-1
14100 IF NXT<=LAST THEN 14120
14110 GOSUB 13000
14120 CARD=DECK(NXT)
14130 T=T+1
14140 NXT=NXT+1
14150 PRINT :"Dealer draws the";
14160 GOSUB 11000
14170 GOSUB 10000
14180 IF CARD<>11 THEN 14200
14190 DACES=DACES+1
14200 DVAL=DVAL+CARD
14210 GOTO 14050
14220 IF DVAL<22 THEN 14280
```

Listing 2-2—cont. Blackjack Program

```
14230 IF DACES=0 THEN 14270
14240 DACES=DACES-1
14250 DVAL=DVAL-10
14260 GOTO 14050
14270 PRINT : :"Dealer's busted."
14280 PRINT " with a total of ";DVAL
14290 FOR I=1 TO ROUND
14300 PRINT "You ";
14310 IF PVAL(I)<>0 THEN 14350
14320 PRINT "LOST ";
14330 MONEY=MONEY-BETS(I)
14340 GOTO 14430
14350 IF DVAL<22 THEN 14390
14360 PRINT "WON ";
14370 MONEY=MONEY+BETS(I)
14380 GOTO 14430
14390 IF DVAL<>PVAL(I)THEN 14420
14400 PRINT "Pushed on ";
14410 GOTO 14430
14420 IF DVAL<PVAL(I)THEN 14360 ELSE 14320
14430 IF ROUND<>1 THEN 14460
14440 PRINT "the hand."
14450 GOTO 14470
14460 PRINT "hand ";I
14470 NEXT I
15000 REM PRINT THE PLAYERS WON/LOST STANDING
15010 PRINT :"You're ";
15020 IF MONEY<>0 THEN 15050
15030 PRINT "EVEN."
15040 RETURN
15050 IF MONEY<0 THEN 15060 ELSE 15080
15060 PRINT "BEHIND $";MONEY
15070 RETURN
15080 PRINT "AHEAD $";MONEY
15090 RETURN
16000 REM INSURANCE ROUTINE
16010 INPUT "Insurance bet(Y/N)->";Y$
16020 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")
      THEN 16110
16030 IF (SEG$(Y$,1,1)<>"Y")*(SEG$(Y$,1,1)<>"y")
      THEN 16010
16040 PRINT "Your insurance bet ";
16050 IF DVAL<>21 THEN 16090
16060 PRINT "WINS."
16070 MONEY=MONEY+BET
16080 RETURN
16090 PRINT "LOSES."
16100 MONEY=MONEY-BET/2
16110 RETURN
17000 REM END OF GAME
17010 PRINT "It was fun.": :
      "Hope you enjoyed yourself."
17020 PRINT : :"Here's your final score!": :
17030 GOSUB 15000
17040 IF MONEY<=0 THEN 17070
17050 PRINT : : :"Collect YOUR WINNINGS":
      "     from the owner!!": : :
17060 STOP
```

Listing 2-2—cont. Blackjack Program

```
17070 IF MONEY<>0 THEN 17100
17080 PRINT : : :"HO HUM...":" You broke even.": : :
17090 STOP
17100 PRINT : : :"Pay the owner of ":
      " this computer ...":"   OR ELSE!!": : :
17110 STOP
18000 REM INSTRUCTIONS
18010 CALL CLEAR
18020 PRINT "  This BLACKJACK program":
      "enables you to play head-to-"
18030 PRINT "head with the computer,":
      "using the rules of the Las"
18040 PRINT "Vagas strip."
18050 PRINT :"  The cards are dealt from":
      "one 52-card deck."
18060 PRINT "The deck is reshuffled as":
      "required, or you can force a"
18070 PRINT "reshuffle by placing a":"negative bet."
18080 PRINT :"  The dealer must hit":
      "through a total of 16 and"
18090 PRINT "stand on all totals of 17":"and above"
18100 GOSUB 19000
18110 PRINT "  The insurance bet is":
      "offered at half the players"
18120 PRINT "bet when the dealer's up-":
      "card is an ace."
18130 PRINT :"  You may double-down on any":
      "two cards, and recieve one"
18140 PRINT "more card; the original":"bet is doubled."
18150 PRINT :"  You may split any hand but":
      "resplitting is not allowed."
18160 GOSUB 19000
18170 CALL CLEAR
18180 PRINT "The commands are as follows:"
18190 PRINT :"  HIT (draw another card)": :
      "  GOOD (stand on current"
18200 PRINT "total)": :"  DOUBLE (double your bet"
18210 PRINT "and draw one card)": :
      "  SPLIT (form two hands from"
18220 PRINT "a pair with the":"original bet on each"
18230 PRINT "hand)": :"  END (ends the game)"
18240 PRINT :"Commands may be typed in as":
      "the first letter."
18250 PRINT "  *** good luck ***": :
18260 GOSUB 19000
18270 CALL CLEAR
18280 RETURN
19000 REM DELAY FOR KEY STROKE
19010 PRINT :TAB(4);"HIT ANY KEY TO GO ON":
19020 CALL KEY(0,K,S)
19030 IF S=0 THEN 19020
19040 RETURN
```

EXTRATERRESTRIAL TYPING PROGRAM

What it does:

You are Head Keeper of the T. Y. P. Qwerty Extraterrestrial Exotic Biology Center. Your job is to care for the nonnative animals that are brought to Earth by the far ranging StarShips.

The ExoBiology exploration and research ship Zoological Gardens has just returned from the newly discovered planet Alpha Betti with a cargo of unusual, and unpredictable, animals native to that planet.

You must keep the animals within the confines of their fenced area. If any escape the area, you must activate the teleportation switch on your control panel that corresponds to the animal that has escaped. This immediately brings that animal back to the fenced area.

But, teleportation equipment has a limited range.

You MUST hit the proper switch BEFORE the animal reaches the mountains.

If you let too many animals escape, YOU'RE FIRED!!!

This is a skill-building game designed to entertain you while you become familiar with the keyboard. Your "control panel" is, of course, the keyboard of your TI-99/TA. The escaping "Alpha Bettis" correspond to the letters, numbers, and punctuation characters inscribed on the keys.

The object is to hit the correct key before the "escapee" makes it safely to the mountains.

You can select from six different levels of difficulty, depending on your skill at the keyboard:

1 Super Expert
2 Touch Typist
3 Two Handed
4 Two Fingered
5 One Fingered
6 Working Blindfolded

Whatever level you select, the program automatically samples your performance every twenty characters. If you're doing well, it makes the game harder either by "moving" the mountains closer or by increasing the speed of the escapees.

If you're not doing so well, it eases up on you by moving the mountains farther away or slowing down the Alpha Bettis.

You can quit playing at any time by hitting the FCTN BACK control key.

When you finish, you can get a list of the number of times each character was presented, the number of misses (hit the wrong key) you had on the character, and the number of times the character escaped from you. You can use this information to see where your keyboard weaknesses lay.

What it shows you:

This program makes heavy use of graphics, both to paint a picture on the screen and to move the Alpha Bettis. It also demonstrates the modular approach to program design.

Look in the 4000 statement number range to see how the keyboard is read without use of an INPUT statement. The 5000 statement number range is also interesting: it is here that the automatic difficulty level adjustment is made.

What you can do:

You could include code to make sure the same character isn't chosen twice in a row. You could also scan the characters array occasionally to force a choice of a character from the LTRS$ array that has not so far been chosen (look in the TRIES array for zero-value entries).

Colors are another choice you can make. The screen background color, the colors chosen for the escaping letters, and the mountain colors are all easily changed to suit your own tastes.

You can also change the selection of characters (see the DATA statements at the end of the program). The double quote (") character is very tough to get in a fast game. Note that the double quote shows up in a DATA statement for four double quotes (""""). You could replace this with, for example, a space.

Table 2-5. Typing Program Routines

Lines	Function
100-300	Initialize variables and screen.
310-490	Main game control loop.
1000-1230	Get player's move.
2000-2120	Sum win vectors; check for winner.
3000-3060	Print the play grid to the screen.
4000-4360	Initialize colors and character patterns.
5000-5070	Place a move on the screen grid.
6000-6900	Computer move logic.
8000-8180	Computer search for winning or blocking move.
9000-9120	Dialog for one or two player game.
20000-20090	Direct print of a message to the screen.
21000-21150	Direct read of a number from the keyboard.

Table 2-6. Typing Program Variables

Variable	Description
ALPH	The character chosen at random to escape.
ASUB	The subscript (in the LTRS$ array) of the randomly chosen character.
COUNT	A running count of the number of characters chosen to escape. Used to determine when to end the game (based on game length).
DELAY	The number of times the keyboard is read for your response. The higher the difficulty level you choose, the fewer times the keyboard is read between each advance of an excapee.
DIR	The direction of movement.
ESCAPEES	The total number of characters that escaped.
ESCAPES(41)	Count of the number of escapes by individual character (corresponds to the LTRS$ array).
GOTIT	A flag set to one (1) when the correct key is pressed.
LEV	The difficulty level you chose (1 to 6).
LSTIMER	The timer value at the last performance check.
LTRS$(41)	Contains the letters, numbers, and punctuation characters chosen as escapees.
MAXESC	The maximum number of escapees allowed at the selected difficulty level.
MAXT	The number of characters to play as determined by the game length.
MCHAR	The mountain drawing character (alters in value between 136 and 137).
MISSES(41)	Count of the number of misses by individual character (corresponds to the LTRS$ array).
NEXTEST	The number of characters played when the next performance test is performed.
NEXTLEV(6)	Contains a difficulty factor for each level that is used to determine when the game should become more or less challenging.
PFACT	A performance factor used to determine whether the game should be made harder, easier, or left as is.
SC	The screen color.
STRIKE	The value of the character last hit on the keyboard.
TDLY	A local count of the number of times the keyboard was read.
TIMER	A running total of the number of chances you've had to hit the correct key. Used with COUNT to determine your overall score.
TRIES(41)	The number of times a character (from the LTRS$ array) was made an escapee.

Listing 2-3. Extraterrestrial Typing Program

```
100 REM THIS IS A TYPING IMPROVEMENT PROGRAM
110 DEF RD(X)=INT(X*100+.5)/100
120 DIM LTRS$(41),MISSES(41),ESCAPES(41),TRIES(41)
130 RANDOMIZE
140 CALL CLEAR
150 GOSUB 1000
160 ASUB=INT(RND*42)
170 ALPH=ASC(LTRS$(ASUB))
180 TRIES(ASUB)=TRIES(ASUB)+1
190 DIR=INT(RND*4)+1
200 COUNT=COUNT+1
210 IF COUNT<>NEXTEST THEN 230
220 GOSUB 5000
230 CALL HCHAR(CROW,CCOL,ALPH)
240 TDLY=0
250 CALL SOUND(-150,1760,0)
260 ON DIR GOSUB 2000,2500,3000,3500
270 IF STRIKE=15 THEN 290
280 IF (COUNT<MAXT)*(ESCAPEES<MAXESC)THEN 160
290 CALL CLEAR
300 IF ESCAPEES<MAXESC THEN 350
310 PRINT "You have allowed ";ESCAPEES:
    " DANGEROUS Alpha Bettis"
320 PRINT " to escape.": :"This is a TERRIBLE record":
    " for a person"
330 PRINT " in your position!!!"
340 PRINT : :"Pack up your computer!!!":
    "  YOU'RE FIRED!!!"
350 PRINT : :"Your average score is ";RD(TIMER/COUNT)
360 GOSUB 7000
370 PRINT : :"Try again?(Y/N)";
380 CALL KEY(0,ALPH,ST)
390 IF ST=0 THEN 380
400 PRINT CHR$(ALPH)
410 IF CHR$(ALPH)="Y" THEN 150
420 IF CHR$(ALPH)<>"N" THEN 370
430 STOP
1000 TIMER=0
1010 COUNT=0
1020 ESCAPEES=0
1030 LSTIMER=0
1040 SC=5
1050 NEXTEST=20
1060 LROW=2
1070 HROW=22
1080 LCOL=4
1090 HCOL=28
1100 FOR I=0 TO 41
1110 TRIES(I)=0
1120 MISSES(I)=0
1130 ESCAPES(I)=0
1140 NEXT I
1150 IF LTRS$(0)="A" THEN 1610
1160 NEXTLEV(1)=10
1170 NEXTLEV(2)=12
1180 NEXTLEV(3)=14
1190 NEXTLEV(4)=16
1200 NEXTLEV(5)=18
```

Listing 2-3—cont. Extraterrestrial Typing Program

```
1210 NEXTLEV(6)=20
1220 CCOL=16
1230 CROW=12
1240 CALL CHAR(128,"017F4549516141FF")
1250 CALL CHAR(129,"81C3A59999A5C381")
1260 CALL CHAR(130,"80FEC2A2928A86FF")
1270 CALL CHAR(131,"FF4224181824 42FF")
1280 CALL CHAR(132,"FF61514945437F01")
1290 CALL CHAR(133,"FF868A92A2C2FE80")
1300 CALL CHAR(136,"0102040813244893")
1310 CALL CHAR(137,"0080403010CC2211")
1320 CALL SCREEN(SC)
1330 FOR I=1 TO 12
1340 CALL COLOR(I,2,3)
1350 NEXT I
1360 CALL COLOR(13,12,2)
1370 CALL COLOR(14,15,7)
1380 FOR I=0 TO 41
1390 READ LTRS$(I)
1400 NEXT I
1410 PRINT : : : : :
1420 PRINT "You are Head Keeper of the":
     "T. Y. P. Qwerty Extraterres-"
1430 PRINT "trial Exotic Biology Center."
1440 PRINT "Your job is to care for the":
     "non-native animals that are"
1450 PRINT "brought to Earth by the far":
     "ranging StarShips."
1460 PRINT :"The ExoBiology exploration":
     "and research ship Zooologi-"
1470 PRINT "cal Gardens has just return-":
     "ed from the newly discovered"
1480 PRINT "planet Alpha Betti with a":
     "cargo of unusual, and unpre-"
1490 PRINT "dictable, animals native to":
     "that planet.": :
1500 GOSUB 8000
1510 PRINT : : :"You must keep the animals":
     "within the confines of their"
1520 PRINT "fenced area.  If any escape":
     "the area, you must activate"
1530 PRINT "the teleportation switch on":
     "your control panel that cor-"
1540 PRINT "responds to the animal that":
     "has escaped. This immediate-"
1550 PRINT "ly brings that animal back":
     "to the fenced area."
1560 PRINT :"BUT, teleportation equip-":
     "ment has a limited range."
1570 PRINT "You MUST hit the proper"
1580 PRINT "switch BEFORE the animal":
     "reaches the mountains."
1590 PRINT "If you let too many animals":
     "escape, YOU'RE FIRED!!!": :
1600 GOSUB 8000
1610 CALL CLEAR
1620 CALL HCHAR(23,1,129,32)
1630 PRINT :"Make sure the ALPHA LOCK key":"is DOWN!!!"
```

Listing 2-3—cont. Extraterrestrial Typing Program

```
1640 PRINT :" "
1650 CALL HCHAR(23,3,129,28)
1660 PRINT : :
1670 PRINT " 1 Super Expert":" 2 Touch Typist":
     " 3 Two Handed"
1680 PRINT " 4 Two Fingered":" 5 One Fingered":
     " 6 Working Blindfolded"
1690 PRINT :"Select your level -> ";
1700 CALL KEY(0,LEV,ST)
1710 IF ST=0 THEN 1700
1720 IF (LEV<49)+(LEV>54)THEN 1660
1730 PRINT CHR$(LEV)
1740 LEV=LEV-48
1750 DELAY=LEV
1760 PRINT : :" 1 Short Game":" 2 Medium Game":
     " 3 Long Game"
1770 PRINT :"Select game length -> ";
1780 CALL KEY(0,MAXT,ST)
1790 IF ST=0 THEN 1780
1800 IF (MAXT<49)+(MAXT>51)THEN 1760
1810 PRINT CHR$(MAXT)
1820 MAXT=MAXT-48
1830 MAXT=MAXT*75
1840 MAXESC=INT(LEV*MAXT/10)
1850 FOR I=1 TO 24
1860 CALL HCHAR(I,1,0,32)
1870 NEXT I
1880 CALL HCHAR(11,15,128)
1890 CALL HCHAR(11,16,129)
1900 CALL HCHAR(11,17,130)
1910 CALL HCHAR(12,15,131)
1920 CALL HCHAR(12,17,131)
1930 CALL HCHAR(13,15,132)
1940 CALL HCHAR(13,16,129)
1950 CALL HCHAR(13,17,133)
1960 CALL HCHAR(12,16,32)
1970 GOSUB 6000
1980 RETURN
2000 CALL HCHAR(CROW,CCOL,32)
2010 GOTIT=0
2020 ROW=10
2030 CALL HCHAR(10,16,ALPH)
2040 GOSUB 4000
2050 IF GOTIT THEN 2180
2060 FOR ROW=9 TO LROW STEP -1
2070 CALL HCHAR(ROW+1,16,0)
2080 CALL HCHAR(ROW,16,ALPH)
2090 GOSUB 4000
2100 IF GOTIT THEN 2180
2110 NEXT ROW
2120 ESCAPEES=ESCAPEES+1
2130 ESCAPES(ASUB)=ESCAPES(ASUB)+1
2140 TIMER=TIMER+50
2150 CALL HCHAR(LROW,16,0)
2160 CALL SOUND(300,131,2,262,4,-2,3)
2170 RETURN
2180 CALL HCHAR(ROW,16,0)
2190 CALL SOUND(120,440,5,1047,0,1319,3,-1,7)
```

Listing 2-3—cont. Extraterrestrial Program

```
2200 RETURN
2500 CALL HCHAR(CROW,CCOL,32)
2510 GOTIT=0
2520 ROW=14
2530 CALL HCHAR(14,16,ALPH)
2540 GOSUB 4000
2550 IF GOTIT THEN 2680
2560 FOR ROW=15 TO HROW
2570 CALL HCHAR(ROW-1,16,0)
2580 CALL HCHAR(ROW,16,ALPH)
2590 GOSUB 4000
2600 IF GOTIT THEN 2680
2610 NEXT ROW
2620 ESCAPEES=ESCAPEES+1
2630 ESCAPES(ASUB)=ESCAPES(ASUB)+1
2640 TIMER=TIMER+50
2650 CALL HCHAR(HROW,16,0)
2660 CALL SOUND(300,131,2,262,4,-2,3)
2670 RETURN
2680 CALL HCHAR(ROW,16,0)
2690 CALL SOUND(120,440,5,1047,0,1319,3,-1,7)
2700 RETURN
3000 CALL HCHAR(CROW,CCOL,32)
3010 GOTIT=0
3020 COL=14
3030 CALL HCHAR(12,14,ALPH)
3040 GOSUB 4000
3050 IF GOTIT THEN 3180
3060 FOR COL=13 TO LCOL STEP -1
3070 CALL HCHAR(12,COL+1,0)
3080 CALL HCHAR(12,COL,ALPH)
3090 GOSUB 4000
3100 IF GOTIT THEN 3180
3110 NEXT COL
3120 ESCAPEES=ESCAPEES+1
3130 ESCAPES(ASUB)=ESCAPES(ASUB)+1
3140 TIMER=TIMER+50
3150 CALL HCHAR(12,LCOL,0)
3160 CALL SOUND(300,131,2,262,4,-2,3)
3170 RETURN
3180 CALL HCHAR(12,COL,0)
3190 CALL SOUND(120,440,5,1047,0,1319,3,-1,7)
3200 RETURN
3500 CALL HCHAR(CROW,CCOL,32)
3510 GOTIT=0
3520 COL=18
3530 CALL HCHAR(12,18,ALPH)
3540 GOSUB 4000
3550 IF GOTIT THEN 3680
3560 FOR COL=19 TO HCOL
3570 CALL HCHAR(12,COL-1,0)
3580 CALL HCHAR(12,COL,ALPH)
3590 GOSUB 4000
3600 IF GOTIT THEN 3680
3610 NEXT COL
3620 ESCAPEES=ESCAPEES+1
3630 ESCAPES(ASUB)=ESCAPES(ASUB)+1
3640 TIMER=TIMER+50
```

Listing 2-3—cont. Extraterrestrial Program

```
3650 CALL HCHAR(12,HCOL,0)
3660 CALL SOUND(300,131,2,262,4,-2,3)
3670 RETURN
3680 CALL HCHAR(12,COL,0)
3690 CALL SOUND(120,440,5,1047,0,1319,3,-1,7)
3700 RETURN
4000 CALL KEY(0,STRIKE,ST)
4010 IF ST<>0 THEN 4040
4020 TDLY=TDLY+1
4030 IF TDLY=DELAY THEN 4150 ELSE 4000
4040 IF STRIKE=15 THEN 4130
4050 IF STRIKE=ALPH THEN 4110
4060 CALL SOUND(-150,-3,0)
4070 MISSES(ASUB)=MISSES(ASUB)+1
4080 TIMER=TIMER+5
4090 TDLY=TDLY+1
4100 GOTO 4030
4110 TIMER=TIMER+TDLY
4120 TDLY=0
4130 GOTIT=1
4140 RETURN
4150 TIMER=TIMER+TDLY
4160 TDLY=0
4170 RETURN
5000 PFACT=(TIMER-LSTIMER)/20
5010 RANDOMIZE
5020 LSTIMER=TIMER
5030 IF PFACT<NEXTLEV(LEV)THEN 5170
5040 IF PFACT<NEXTLEV(LEV)*1.3 THEN 5260
5050 IF HROW=22 THEN 5150
5060 HROW=HROW+1
5070 LROW=LROW-1
5080 HCOL=HCOL+1
5090 LCOL=LCOL-1
5100 CALL HCHAR(LROW,LCOL,0,HCOL-LCOL)
5110 CALL HCHAR(HROW,LCOL,0,HCOL-LCOL)
5120 CALL VCHAR(LROW,LCOL,0,HROW-LROW)
5130 CALL VCHAR(LROW,HCOL,0,HROW-LROW+1)
5140 GOTO 5260
5150 DELAY=DELAY+1
5160 GOTO 5260
5170 IF HROW=15 THEN 5240
5180 HROW=HROW-1
5190 LROW=LROW+1
5200 HCOL=HCOL-1
5210 LCOL=LCOL+1
5220 GOSUB 6000
5230 GOTO 5260
5240 IF DELAY=1 THEN 5260
5250 DELAY=DELAY-1
5260 NEXTEST=COUNT+20
5270 RETURN
6000 FOR I=1 TO LCOL-1 STEP 2
6010 CALL VCHAR(1,I,136,24)
6020 NEXT I
6030 FOR I=2 TO LCOL-1 STEP 2
6040 CALL VCHAR(1,I,137,24)
6050 NEXT I
```

Listing 2-3—cont. Extraterrestrial Typing Program

```
6060 MCHAR=137
6070 IF INT(LCOL/2)*2=LCOL THEN 6090
6080 MCHAR=136
6090 FOR I=LCOL TO HCOL
6100 CALL VCHAR(1,I,MCHAR,LROW-1)
6110 CALL VCHAR(HROW+1,I,MCHAR,24-HROW)
6120 MCHAR=273-MCHAR
6130 NEXT I
6140 MCHAR=136
6150 IF INT((HCOL+1)/2)*2=HCOL THEN 6170
6160 MCHAR=137
6170 FOR I=HCOL+1 TO 32
6180 CALL VCHAR(1,I,MCHAR,24)
6190 MCHAR=273-MCHAR
6200 NEXT I
6210 RETURN
7000 PRINT : :"Do you want to see":
     " your detailed score? (Y/N)";
7010 CALL KEY(0,ALPH,ST)
7020 IF ST=0 THEN 7010
7030 PRINT CHR$(ALPH)
7040 IF CHR$(ALPH)="Y" THEN 7070
7050 IF CHR$(ALPH)<>"N" THEN 7000
7060 RETURN
7070 GOSUB 7180
7080 FOR I=0 TO 20
7090 PRINT TAB(3);LTRS$(I);TAB(8);TRIES(I);TAB(14);
     MISSES(I);TAB(22);ESCAPES(I)
7100 NEXT I
7110 GOSUB 8000
7120 GOSUB 7180
7130 FOR I=21 TO 41
7140 PRINT TAB(3);LTRS$(I);TAB(8);TRIES(I);TAB(14);
     MISSES(I);TAB(22);ESCAPES(I)
7150 NEXT I
7160 GOSUB 8000
7170 RETURN
7180 CALL CLEAR
7190 PRINT "LETTER";TAB(8);"TRIES";TAB(14);"MISSES";
     TAB(21);"ESCAPES"
7200 RETURN
8000 PRINT "HIT ANY KEY TO CONTINUE."
8010 CALL KEY(0,ALPH,ST)
8020 IF ST=0 THEN 8010
8030 RETURN
10000 DATA A,B,C,D,E,F,G,H,I
10020 DATA J,K,L,M,N,O,P,Q,R
10040 DATA S,T,U,V,W,X,Y,Z
10060 DATA ",",.,;,0,1,2,3,4
10080 DATA 5,6,7,8,9,/,"""",=
```

PLANET FALL PROGRAM

What it does:

The Planet Fall is a space probe landing simulator program. You are randomly routed to one of five destinations with varying gravity, initial speed, and fuel load.

At each second of your descent, you are asked for the amount of fuel you wish to burn. After the burn, your speed, altitude, fuel remaining, and time to impact are displayed on your control panel. In addition to the digital display, there is a graphic display showing fuel remaining and ship's altitude.

What it shows you:

There is heavy use of graphics in the control of the fuel gauge and rocket altitude displays. If you're interested in fine graphics control in TI BASIC, the routines in the 6000 and 7000 statement number range use some techniques which you can adapt.

Another interesting aspect of this program is the direct positioning input/output routines in the 20000 and 21000 statement number range. These routines simulate some of the behavior of the DISPLAY AT and ACCEPT FROM Extended BASIC statements.

What you can do:

It's relatively easy to change the color combinations we've chosen for the characters and screen for each landing site. If you prefer a different set of colors, by all means make things the colors that you like (see statements 2090–2520).

We've given you plenty of fuel to make an easy landing. To make the game harder, reduce the initial fuel allotment (statement 2020).

It's very easy to add new planets to those we've invented, or to replace the ones we have. Be creative! Invent your own planets, names, conditions, and colors.

Table 2-7. Planet Fall Routines

Lines	Function
100-370	Introductory code and initialization control.
380-940	Main game control loop and game end.
1000-1230	Get amount of fuel to burn.
2000-2530	Randomly select a target planet and set up colors, gravity, fuel, initial velocity, and altitude.
3000-3290	Print the rules of the game.
4000-4260	Print the initial screen display.
5000-5230	Update the numeric data on the screen.
6000-6180	Update the fuel gauge.
7000-7390	Update the rocket display.
8000-8210	Initialize system variables.
9000-9040	Delay routine.
10000-10220	Rocket crash landing siren and crash sounds.
20000-20090	Direct output routine.
21000-21300	Direct numeric input routine.

Table 2-8. Planet Fall Variables

Variable	Description
ANS	Numeric input routine answer.
BURN	Amount of fuel to burn.
CCOLOR	Character color.
FBCOLOR	Fuel gauge background color.
FDEF$(8)	Fuel gauge graphic display characters.
FGCOLOR	Fuel gauge foreground color.
FINX	Index into the FDEF$ array.
FPIXEL	The current fuel gauge pixel.
FPOS	Currently active fuel gauge screen column.
FUEL	Amount of fuel remaining.
G	Gravity in meters/sec/sec.
HT	Height.
MAXTHRUST	Maximum amount of fuel you can burn.
OFUEL	Original fuel allocation.
OHT	Original height.
OUTPAT$	Rocket graphic display motion pattern.
PLANET$	Planet name.
RBCOLOR	Rocket display background color.
RCOLOR	Rocket color.
RKPIX	Current rocket pixel.
RKROW	Currently active rocket display row.
RPAT$	Rocket pattern string.
SCCOLOR	Screen color.
TIME	Elapsed landing time in seconds.
VEL	Current velocity.
ZEROS$	Blank character pattern string.
USER FUNCTIONS	
FILL$(X)	Pads MSG$ with blanks to a constant length as given by X.
RD(X)	Rounds X to two digits past the decimal (nn.nn).

Listing 2-4. Planet Fall Program

```
100 REM PLANET FALL LANDING
110 REM SIMULATOR.
120 DEF RD(X)=INT(X*100)/100
130 DEF FILL$(X)=MSG$&SEG$("          ",1,X-LEN(MSG$))
140 CALL CLEAR
150 RANDOMIZE
160 PRINT TAB(6);"PROBE SHIP SS993"
170 PRINT :TAB(7);"MANUAL LANDING"
180 PRINT TAB(5);"SEQUENCE INITIATED"
190 PRINT TAB(5);"-----------------"
200 PRINT : : : : : : : :
210 GOSUB 9000
220 GOSUB 8000
230 GOSUB 2000
240 PRINT :"DO YOU WANT INSTRUCTIONS"
250 INPUT "  (YES OR NO) ":A$
260 IF SEG$(A$,1,1)="Y" THEN 300
```

Listing 2-4—cont. Planet Fall Program

```
270 IF SEG$(A$,1,1)="N" THEN 370
280 PRINT "YES OR NO"
290 GOTO 240
300 GOSUB 3000
310 PRINT :"DO YOU UNDERSTAND THE"
320 INPUT " INSTRUCTIONS? ":A$
330 IF SEG$(A$,1,1)="Y" THEN 370
340 IF SEG$(A$,1,1)="N" THEN 300
350 PRINT "YES OR NO "
360 GOTO 310
370 CALL CLEAR
380 PRINT " APPROACHING TARGET PLANET": :
    " YOU HAVE MANUAL CONTROL!"
390 PRINT : : : : : : : :
400 FOR I=1 TO 200
410 NEXT I
420 GOSUB 4000
430 GOSUB 1000
440 NEWVEL=VEL-BURN+G
450 FUEL=FUEL-BURN
460 HT=HT-.5*(VEL+NEWVEL)
470 IF HT<=0 THEN 610
480 TIME=TIME+1
490 VEL=NEWVEL
500 GOSUB 5000
510 GOSUB 6000
520 GOSUB 7000
530 IF FUEL>0 THEN 430
540 IF BURN=0 THEN 440
550 MSG$="*** OUT OF FUEL *** "
560 MROW=16
570 MCOL=1
580 GOSUB 20000
590 BURN=0
600 GOTO 440
610 CALL CLEAR
620 HT=HT+.5*(VEL+NEWVEL)
630 IF BURN=G THEN 660
640 DFACTOR=(-VEL+SQR(VEL*VEL+HT*(2*G-2*BURN)))/
    (G-BURN)
650 GOTO 670
660 DFACTOR=HT/VEL
670 NEWVEL=VEL+(G-BURN)*DFACTOR
680 IF NEWVEL>2 THEN 720
690 PRINT TAB(6);"CONGRATULATIONS!!":TAB(5);
    "A PERFECT LANDING!"
700 PRINT : :"Your license is, of course,":" renewed."
    : : :"We are offering a bonus of":
    "1400 Venadian Blots (veri-"
710 PRINT "fied) for re-enlisting.": :
    "We do hope you'll consider":
    " this generous offer."
720 IF ABS(NEWVEL)<4 THEN 830
730 GOSUB 10000
740 PRINT "PROBE SHIP S993 DESTROYED ON"
750 PRINT "LANDING DUE TO PILOT ERROR."
760 PRINT : :"Unfortunately,":"irregularities in the"
770 PRINT "insurance policy will delay"
```

Listing 2-4—cont. Planet Fall Program

```
780 PRINT "payment of survivor's"
790 PRINT "benefits..."
800 FOR I=1 TO 50
810 NEXT I
820 PRINT : :"Indefinitely."
830 PRINT : : : :
840 PRINT "TOUCHDOWN AT";RD(TIME+DFACTOR);"SECONDS."
850 PRINT "LANDING VELOCITY =";RD(NEWVEL);"M/SEC"
860 PRINT RD(FUEL);" UNITS OF FUEL REMAINING."
870 PRINT : :
880 INPUT "ANOTHER MISSION? ":A$
890 IF SEG$(A$,1,1)="N" THEN 930
900 IF SEG$(A$,1,1)="Y" THEN 220
910 PRINT "YES OR NO"
920 GOTO 870
930 PRINT :"MISSION COMPLETE.": :" "
940 STOP
1000 REM GET FUEL TO BURN
1010 MROW=16
1020 MCOL=17
1030 GOSUB 21000
1040 BURN=ANS
1050 IF BURN<=MAXTHRUST THEN 1180
1060 MSG$="MAXIMUM BURN IS "&STR$(MAXTHRUST)
1070 MROW=18
1080 MCOL=3
1090 GOSUB 20000
1100 CALL SOUND(300,131,2,262,4,-2,3)
1110 FOR K=1 TO 50
1120 NEXT K
1130 MCOL=3+LEN(MSG$)
1140 MSG$=""
1150 OCOL=3
1160 GOSUB 20000
1170 GOTO 1010
1180 IF BURN>0 THEN 1210
1190 BURN=0
1200 RETURN
1210 IF BURN<FUEL THEN 1230
1220 BURN=FUEL
1230 RETURN
2000 REM SELECT PLANET AT RANDOM
2010 ON INT(RND*5+1)GOSUB 2090,2180,2270,2360,2450
2020 FUEL=INT((50+(RND*7))*G)
2030 OFUEL=FUEL
2040 HT=100*G
2050 OHT=HT
2060 MAXTHRUST=INT(6*G)
2070 VEL=INT(HT/(8*RND*10))
2080 RETURN
2090 PLANET$="MARS"
2100 G=3.72
2110 SCCOLOR=6
2120 CCOLOR=5
2130 RCOLOR=2
2140 RBCOLOR=15
2150 FGCOLOR=7
2160 FBCOLOR=12
```

Listing 2-4—cont. Planet Fall Program

```
2170 RETURN
2180 PLANET$="CYGNUS X-1"
2190 G=1253
2200 SCCOLOR=2
2210 CCOLOR=11
2220 RCOLOR=7
2230 RBCOLOR=15
2240 FGCOLOR=3
2250 FBCOLOR=16
2260 RETURN
2270 PLANET$="TITAN"
2280 G=3.24
2290 SCCOLOR=13
2300 CCOLOR=2
2310 RCOLOR=2
2320 RBCOLOR=8
2330 FGCOLOR=5
2340 FBCOLOR=16
2350 RETURN
2360 PLANET$="VENUS"
2370 G=8.6
2380 SCCOLOR=15
2390 CCOLOR=13
2400 RCOLOR=13
2410 RBCOLOR=11
2420 FGCOLOR=2
2430 FBCOLOR=7
2440 RETURN
2450 PLANET$="BARNARD IV"
2460 G=9.78
2470 SCCOLOR=6
2480 CCOLOR=2
2490 RCOLOR=7
2500 RBCOLOR=15
2510 FGCOLOR=2
2520 FBCOLOR=16
2530 RETURN
3000 REM EXPLAIN THE RULES
3010 PRINT : : :"You are landing your space"
3020 PRINT "probe on ";PLANET$;" under"
3030 PRINT "manual control.  You take"
3040 PRINT "control at an altitude of"
3050 PRINT HT;" meters traveling at"
3060 PRINT VEL;" meters/second with"
3070 PRINT FUEL;" kilos of fuel"
3080 PRINT "remaining."
3090 PRINT :"Each second of your descent"
3100 PRINT "you specify the amount of"
3110 PRINT "fuel you want to burn to"
3120 PRINT "a maximum of ";MAXTHRUST;"."
3130 PRINT :"Don't burn too much fuel or"
3140 PRINT "you'll go up -- not down!"
3150 GOSUB 9000
3160 PRINT :"You must impact the surface"
3170 PRINT "at less than 2 m/sec to"
3180 PRINT "retain your pilot's"
3190 PRINT "license."
3200 PRINT : :"If you reach the surface"
```

Listing 2-4—cont. Planet Fall Program

```
3210 PRINT "travelling at more than 4"
3220 PRINT "m/sec your ship is"
3230 PRINT "destroyed."
3240 FOR I=1 TO 300
3250 NEXT I
3260 PRINT : : :"Oh. And so are you."
3270 PRINT : :
3280 GOSUB 9000
3290 RETURN
4000 REM SET UP SCREEN DISPLAY
4010 CALL CLEAR
4020 CALL SCREEN(SCCOLOR)
4030 FOR I=1 TO 12
4040 CALL COLOR(I,CCOLOR,SCCOLOR)
4050 NEXT I
4060 CALL COLOR(13,RCOLOR,RBCOLOR)
4070 CALL COLOR(14,FGCOLOR,FBCOLOR)
4080 CALL CHAR(128,"1818183C7EFFDB99")
4090 CALL CHAR(129,"0000000000000000")
4100 CALL CHAR(130,"0000000000000000")
4110 CALL CHAR(136,"0000000000000000")
4120 CALL CHAR(137,"FFFFFFFFFFFFFFFF")
4130 CALL CHAR(138,"FFFFFFFFFFFFFFFF")
4140 PRINT TAB(11);"FUEL": : :PLANET$;TAB(13);
     "GRAVITY:";STR$(G)
4150 PRINT :"FUEL REMAINING:";STR$(FUEL)
4160 PRINT :"DESCENT RATE: ";STR$(VEL)
4170 PRINT :"TIME TO IMPACT:";STR$(RD(HT/VEL))
4180 PRINT :"ALTITUDE:";STR$(HT)
4190 PRINT :"FUEL TO BURN->"
4200 PRINT : : : : : : :
4210 CALL VCHAR(5,28,130,15)
4220 CALL VCHAR(3,28,128)
4230 CALL VCHAR(4,28,129)
4240 CALL HCHAR(4,3,137,19)
4250 CALL HCHAR(4,22,138)
4260 RETURN
5000 REM UPDATE THE NUMERIC DISPLAY
5010 MSG$=STR$(FUEL)
5020 MSG$=FILL$(5)
5030 MROW=8
5040 MCOL=17
5050 GOSUB 20000
5060 MSG$=STR$(RD(VEL))
5070 MSG$=FILL$(6)
5080 MROW=10
5090 MCOL=15
5100 GOSUB 20000
5110 MSG$="***"
5120 IF VEL<=0 THEN 5140
5130 MSG$=STR$(RD(HT/VEL))
5140 MSG$=FILL$(8)
5150 MROW=12
5160 MCOL=17
5170 GOSUB 20000
5180 MSG$=STR$(RD(HT))
5190 MSG$=FILL$(8)
5200 MROW=14
```

Listing 2-4—cont. Planet Fall Program

```
5210 MCOL=11
5220 GOSUB 20000
5230 RETURN
6000 REM UPDATE FUEL GUAGE
6010 IF FPIXEL=0 THEN 6180
6020 NEWPIX=INT(FUEL/OFUEL*160)
6030 IF NEWPIX=FPIXEL THEN 6180
6040 K=FPIXEL-NEWPIX
6050 FPIXEL=NEWPIX
6060 FOR J=1 TO K
6070 FINX=FINX-1
6080 IF FINX<>0 THEN 6160
6090 CALL HCHAR(4,FPOS,136)
6100 FPOS=FPOS-1
6110 FINX=8
6120 IF FPOS=2 THEN 6180
6130 CALL CHAR(138,FDEF$(8))
6140 CALL HCHAR(4,FPOS,138)
6150 GOTO 6170
6160 CALL CHAR(138,FDEF$(FINX))
6170 NEXT J
6180 RETURN
7000 REM MOVE THE ROCKET
7010 NEWPIX=INT(HT/OHT*128)
7020 IF NEWPIX=RKPIX THEN 7390
7030 IF NEWPIX>RKPIX THEN 7230
7040 IF (NEWPIX>127)+(NEWPIX<1)THEN 7200
7050 REM MOVE THE ROCKET DOWN
7060 FOR CURPIX=RKPIX-1 TO NEWPIX STEP -1
7070 IF CURPIX/8<>INT(CURPIX/8)THEN 7160
7080 RKROW=RKROW+1
7090 OUTPAT$=RPAT$&ZEROS$
7100 CALL CHAR(128,RPAT$)
7110 CALL VCHAR(RKROW,28,128)
7120 CALL VCHAR(RKROW-1,28,130)
7130 CALL CHAR(129,ZEROS$)
7140 CALL VCHAR(RKROW+1,28,129)
7150 GOTO 7190
7160 OUTPAT$=SEG$("00"&OUTPAT$,1,32)
7170 CALL CHAR(128,OUTPAT$)
7180 CALL CHAR(129,SEG$(OUTPAT$,17,16))
7190 NEXT CURPIX
7200 RKPIX=NEWPIX
7210 RETURN
7220 REM MOVE THE ROCKET UP
7230 FOR CURPIX=RKPIX+1 TO NEWPIX
7240 IF CURPIX>127 THEN 7380
7250 IF CURPIX/8<>INT(CURPIX/8)THEN 7340
7260 RKROW=RKROW-1
7270 OUTPAT$=ZEROS$&RPAT$
7280 CALL CHAR(128,ZEROS$)
7290 CALL VCHAR(RKROW,28,128)
7300 CALL VCHAR(RKROW+2,28,130)
7310 CALL CHAR(129,RPAT$)
7320 CALL VCHAR(RKROW+1,28,129)
7330 GOTO 7370
7340 OUTPAT$=SEG$(OUTPAT$,3,30)&"00"
7350 CALL CHAR(128,OUTPAT$)
```

Listing 2-4—cont. Planet Fall Program

```
7360 CALL CHAR(129,SEG$(OUTPAT$,17,16))
7370 NEXT CURPIX
7380 RKPIX=NEWPIX
7390 RETURN
8000 REM INIT SYSTEM VARIABLES
8010 FPIXEL=160
8020 FPOS=22
8030 FINX=8
8040 RKROW=3
8050 RKPIX=128
8060 RPAT$="1818183C7EFFDB99"
8070 ZEROS$="0000000000000000"
8080 OUTPAT$=RPAT$&ZEROS$
8090 RESTORE
8100 FOR I=1 TO 8
8110 READ FDEF$(I)
8120 NEXT I
8130 RETURN
8140 DATA 8080808080808080
8150 DATA C0C0C0C0C0C0C0C0
8160 DATA E0E0E0E0E0E0E0E0
8170 DATA F0F0F0F0F0F0F0F0
8180 DATA F8F8F8F8F8F8F8F8
8190 DATA FCFCFCFCFCFCFCFC
8200 DATA FEFEFEFEFEFEFEFE
8210 DATA FFFFFFFFFFFFFFFF
9000 REM DELAY FOR KEY STROKE
9010 PRINT :TAB(4);"HIT ANY KEY TO GO ON":
9020 CALL KEY(0,K,S)
9030 IF S=0 THEN 9020
9040 RETURN
10000 FOR I=1 TO 4
10010 FOR S=500 TO 800 STEP 40
10020 CALL SOUND(-200,S,0,S+75,0)
10030 NEXT S
10040 FOR S=800 TO 500 STEP -40
10050 CALL SOUND(-200,S,0,S+75,0)
10060 NEXT S
10070 NEXT I
10080 FOR S=790 TO 110 STEP -30
10090 A=.0435*(800-S)
10100 CALL SOUND(-200,S,A,S+75,A)
10110 NEXT S
10120 FOR I=1 TO 2
10130 CALL SOUND(33,-5,4)
10140 CALL SOUND(370,-6,0)
10150 CALL SOUND(333,-6,4)
10160 CALL SOUND(303,-6,8)
10170 CALL SOUND(100,-6,12)
10180 CALL SOUND(67,-6,16)
10190 CALL SOUND(33,-7,18)
10200 CALL SOUND(33,-6,18)
10210 NEXT I
10220 RETURN
20000 REM PRINT MSG$ AT MROW STARTING IN MCOL
20010 MLEN=LEN(MSG$)
20020 IF MLEN=0 THEN 20080
20030 FOR OCOL=1 TO MLEN
```

Listing 2-4—cont. Planet Fall Program

```
20040 CALL HCHAR(MROW,OCOL+MCOL,ASC(SEG$(MSG$,OCOL,1)))
20050 NEXT OCOL
20060 MCOL=MCOL+OCOL
20070 RETURN
20080 CALL HCHAR(MROW,OCOL,32,MCOL-OCOL+1)
20090 RETURN
21000 REM GET A NUMERIC ANSWER
21010 FOR OCOL=0 TO 6
21020 CALL HCHAR(MROW,MCOL+OCOL,32)
21030 NEXT OCOL
21040 DIGITS=0
21050 CALL KEY(0,ONE,S)
21060 IF S=0 THEN 21050
21070 CALL HCHAR(MROW,MCOL,ONE)
21080 ONE=ONE-48
21090 IF (ONE>=0)*(ONE<10)THEN 21130
21100 CALL SOUND(-300,131,0,262,2,-2,1)
21110 GOTO 21050
21120 DIGITS=1
21130 MCOL=MCOL+1
21140 ANS=ONE
21150 FOR K=1 TO 20
21160 NEXT K
21170 CALL KEY(0,TWO,S)
21180 IF S=0 THEN 21170
21190 IF TWO=13 THEN 21300
21200 CALL HCHAR(MROW,MCOL,TWO)
21210 TWO=TWO-48
21220 IF (TWO>=0)*(TWO<10)THEN 21250
21230 CALL SOUND(-300,131,0,262,2,-2,1)
21240 GOTO 21170
21250 ANS=ANS*10+TWO
21260 MCOL=MCOL+1
21270 DIGITS=DIGITS+1
21280 IF DIGITS=5 THEN 21300
21290 GOTO 21170
21300 RETURN
```

SPACE WAR 7 PROGRAM

What it does:

Space War 7 is based on Startrek. This is one of the most enduring computer games—and for good reason. It is a rich, exciting, and challenging game of strategy and tactics.

Startrek games were originally written to run under the time-sharing (terminal) systems of large mainframe computers. First written in FORTRAN, versions of the game now exist in many languages. It seems that every computer hacker between the years 1968 and 1975 wrote or made large additions to the game.

As a result of all this tinkering, there are now literally hundreds of

different versions of Startrek, each with its own blend of the features of the early programs and including unique elements invented by the individual developers.

In developing Space War 7 to run in the 16K TI-99/4A, we were careful to maintain the best points of the originals. The major surgery we performed to make it fit in 16K was cutting the size of the Galaxy from an 8 by 8 grid to a 7 by 7, and likewise cutting the size of a quadrant from 10 by 10 to 7 by 7.

Once you've entered the program, RUN it and answer the questions. Be sure to remember your password; you'll need it if you want to self-destruct.

The program will ask your rank. The higher your rank, the more aggressive are the Krinng and the more likely you are to take damage in an attack.

The Galaxy is a 7 by 7 grid of **quadrants.** Each quadant is a 7 by 7 grid of **sectors.** Your position in the Galaxy is expressed by four numbers:

quadrant row, quadrant column, sector row, sector column

So, for example, if you are in the quadrant on the third row and the fourth column of the Galaxy Chart, you are in Quadrant (3,4). If you are in the sector on the fifth row, seventh column of that quadrant, you are in Sector (5,7).

Your function, as Captain of the Enterprise, is to rid the Galaxy of Krinng. You do this by flying from quadrant (one spot in the grid) to quadrant searching for the Krinng and engaging them in battle.

The Galaxy also contains stars and *bases.* You **dock at a base** by maneuvering the Enterprise to a sector adjacent to the base. You must dock at a base to refuel and replenish your supply of photon torpedoes. Also, damage repair works only when you are docked at a base.

When the computer asks you for a command, you can get a **list of valid commands** by entering a large number—like 99.

The primary commands are:

1=SHORT-RANGE SCAN
2=LONG-RANGE SCAN
3=TORPEDOES
4=PHASERS
5=SHIELDS
6=WARP DRIVE
7=COMPUTER

8=DAMAGE REPORT
9=DAMAGE CONTROL

You have two scanners at your disposal. The *Short-Range* scanner shows you the content and location of objects in your quadrant.

The *Long-Range* scanner tells you what's in the quadrants immediately adjacent to yours. This information is presented as follows:

- the **first digit** is the number of **Krinng** in the sector
- the **second digit** is the number of **bases** in the sector
- the **third digit** is the number of **stars** in the sector

Thus the display:

216

indicates that there are two (2) Krinng, one (1) base, and six (6) stars in the sector.

Directions are expressed in terms of a clock face:

- **UP** is the 0 or 12 o'clock direction
- **RIGHT** is the 3 o'clock direction
- **DOWN** is the 6 o'clock direction
- **LEFT** is the 9 o'clock direction

If you're going to use a direction between two numbers, use *decimals* to express it. For example, half way between 4 o'clock and 5 o'clock is 4.5 (*not* 4:30).

You move about using the *Warp Drive*. The *warp factor* is actually the distance you want to move. A warp factor of one moves you one *quadrant*. To move within a quadrant, use warp factors less than one. For example, a warp factor of .3 moves you three sectors in the direction you choose within the same quadrant (unless you cross a quadrant boundary).

You have two kinds of *weapons:* phasers and photon torpedoes. Phasers don't have to be aimed, but they consume valuable energy. Photon torpedoes must be aimed (use the computer to get course information), are limited in number, and are subject to countermeasures from the Krinng. On the other hand, they don't consume energy.

Your *Shields* are your principal defense against Krinng attack. You must assign part of your total energy to your shields. Each Krinng attack depletes the energy in your shields. Watch out for that, or you'll find yourself destroyed because you've allowed your shield energy to sag too low.

If you take damages—and you will—your *Damage Report* tells

you how badly off you are. You can use *Damage Repair* only when you are docked at a base.

The on-board *Computer* contains a second set of commands:

1=CHART
2=STATUS
3=TORPEDO COURSE
4=POSITION
5=DESTRUCT
6=EXIT

The *Chart* command produces a chart of the entire Galaxy, showing all that you currently know of it. If you have not visited or scanned a quadrant, it shows on the chart as three dots (. . .). If you have scanned the quadrant, the chart shows the same information as the long-range scanner.

As you progress, the *Status* command tells you how many Krinng are left, how much time you have, and how you're fixed for energy and torpedoes.

When you are engaged in battle, you can request a *Torpedo Course* report from the computer. This provides you the course information you need to launch torpedoes against the Krinng in your sector. Watch out, though, sometimes they move.

The *Position* command simply tells you where you are. It lists the current quadrant and sector coordinates of the Enterprise.

When all else fails and the Krinng are about to destroy the Enterprise, you can always go out in a blaze of glory. The *Destruct* command provides the honorable way out in the event of defeat. To use it, though, you must remember your password.

What it shows you:

Space War 7 is a very large program that just barely fits into the 16K available in the TI-99/4A console. You will notice that the variable names in this program are very short (one or two characters). We do not normally recommend such short, and obscure, variable names. In this case, though, it was necessary to use them in order to get the program to run at all.

Note the use of *functions* to perform often repeated calculations, such as rounding and computing distances.

What you can do:

Not much, unless you have Memory Expansion and Extended BASIC. There really isn't room for any more code in this program.

Table 2-9. Space War 7 Routines

Lines	Function
100-200	Initial setup.
210-340	Main game control loop.
1000-1090	Short-range scanner.
2000-2150	Long-range scanner.
3000-3590	Fire photon torpedoes.
4000-4340	Fire phasers.
5000-5060	Transfer energy to the shields.
6000-6870	Warp drive.
7000-7070	Computer main control routine.
7080-7210	List Galaxy chart.
7220-7250	List status.
7260-7530	Torpedo course calculator.
7540-7590	List position of Enterprise.
7600-7650	Self-destruct.
8000-8040	Damage report.
9000-9250	Damage repair.
10000-10280	Krinng attack.
11000-11170	End of game.
12000-13500	Game initialization.
14000-14070	Get course.

If you do have more memory on your system, you can improve the messages, making them more realistic, as in:

Engineering to Bridge: "We haven't the energy
to do that, Capt'n."

You can also add sound effects and color to the action. If you're really ambitious, you can add new commands.

Table 2-10. Space War 7 Variables

Variable	Description
B1	Sector coordinate of a base.
B2	Sector coordinate of a base.
BA	Number of bases in current quadrant.
BL	Number of bases left.
C$	Set to "D" when Enterprise is docked at a base.
CS	Course in degrees.
CX	Degree to radian conversion factor.
D(9)	Corresponds to the D$ array. Indicates the extent of damages to a system.
D$(9)	Contains the names of the primary commands.
EN	Energy remaining.
G(7,7)	The Galaxy array. Each cell contains a three-digit number indicating the content of the quadrant.
K(3,3)	Indicates the location of the Krinng in a quadrant.
KL	Krinng remaining in the Galaxy.
KS	Krinng remaining in the quadrant.
MAXSP	Maximum speed (warp).
Q1	Quadrant coordinate of the Enterprise.
Q2	Quadrant coordinate of the Enterprise.
RF	Rank factor used in computations.
RK	Your rank.
S(7,7)	The Quadrant array. Indicates the object at each sector in the quadrant.
S1	Sector coordinate of the Enterprise.
S2	Sector coordinate of the Enterprise.
S3	Alternate sector coordinate.
S4	Alternate sector coordinate.
SC$	Secret destruct code.
SD	Number of stardates to get the Krinng.
SH	Shield energy remaining.
TP	Number of torpedoes remaining.
TU	Stardates used.
U	Original number of Krinng.
WP	Warp factor.
Y	Original number of bases in the Galaxy.
USER FUNCTIONS	
DST	Computes distance from the Enterprise to a Krinng.
DTA(X)	Computes first coordinate of destination sector.
DTB(X)	Computes second coordinate of destination sector.
P2(X)	Rounds to two decimal places.
RD	Random selection of a number between 1 and 7.

Listing 2-5. Space War 7 Program

```
100 OPTION BASE 1
110 DIM GL(7,7),SC(7,7),K(3,3),A$(5),D$(9)
120 DEF DT=SQR((K(I,1)-S1)^2+(K(I,2)-S2)^2)
130 DEF DTA(X)=INT(.5+A1+X*COS(CS*CX))
140 DEF DTB(X)=INT(.5+B1+X*SIN(CS*CX))
150 DEF RD=INT(RND*7+1)
160 DEF P2(X)=INT(X*100)/100
170 RANDOMIZE
180 CALL CLEAR
190 PRINT "SPACE WAR 7": :
200 GOSUB 12000
210 INPUT "COMMAND(1-9)->":A
220 IF (A>9)+(A<1)THEN 300
230 IF A=6 THEN 270
240 IF D(A)>=0 THEN 270
250 PRINT D$(A);" DAMAGED"
260 GOTO 210
270 ON A GOSUB 1000,2000,3000,4000,5000,6000,
    7000,8000,9000
280 PRINT
290 GOTO 210
300 PRINT "COMMANDS"
310 FOR I=1 TO 9
320 PRINT I;"=";D$(I)
330 NEXT I
340 GOTO 210
1000 PRINT D$(1)
1010 FOR A=1 TO 7
1020 FOR B=1 TO 7
1030 PRINT " ";A$(SC(A,B)+1);
1040 NEXT B
1050 PRINT
1060 NEXT A
1070 PRINT :"SHLDS ";P2(SH):"ENRGY ";P2(EN):"TORP ";
     P2(TP):"STARDATES ";P2(SD-TU)
1080 GOSUB 7540
1090 RETURN
2000 PRINT D$(2)
2010 FOR A=(Q1-1)TO (Q1+1)
2020 FOR B=(Q2-1)TO (Q2+1)
2030 IF (A<1)+(B<1)+(A>7)+(B>7)THEN 2040 ELSE 2060
2040 ST$="***"
2050 GOTO 2090
2060 ST$="..."&STR$(INT(GL(A,B)))
2070 ST$=SEG$(ST$,LEN(ST$)-2,3)
2080 GL(A,B)=INT(GL(A,B))+.5
2090 PRINT " ";ST$;
2100 NEXT B
2110 PRINT
2120 NEXT A
2130 PRINT
2140 GOSUB 7540
2150 RETURN
3000 PRINT D$(3);"(";STR$(TP);")":
3010 IF TP>0 THEN 3040
3020 PRINT "NO TORPEDOES"
3030 RETURN
3040 GOSUB 14000
```

Listing 2-5—cont. Space War 7 Program

```
3050 TU=TU+.05
3060 IF TU>SD THEN 11050
3070 TP=TP-1
3080 GOSUB 3110
3090 GOSUB 10000
3100 RETURN
3110 A1=7*Q2+S2-8
3120 B1=56-7*Q1-S1
3130 IF CS=90*INT(CS/90)THEN 3550
3140 FOR WP=0.0 TO 10.0 STEP .9
3150 IF WP<>0 THEN 3190
3160 S3=S1
3170 S4=S2
3180 GOTO 3260
3190 IF (Q1=7-INT(DTB(WP)/7))*(Q2=INT(DTA(WP)/7+1))
     THEN 3210
3200 RETURN
3210 S3=7-DTB(WP)+7*INT(DTB(WP)/7)
3220 S4=DTA(WP)+1-7*INT(DTA(WP)/7)
3230 IF (DTA(WP)=A2)*(DTB(WP)=B2)THEN 3530
3240 A2=DTA(WP)
3250 B2=DTB(WP)
3260 PRINT "(";STR$(S3);",";STR$(S4);") ";
     A$(SC(S3,S4)+1)
3270 ON SC(S3,S4)+1 GOTO 3530,3530,3510,3280,3450
3280 IF RND>=(.04*RK)THEN 3310
3290 PRINT "SHIELDS DEFLECT TORPEDO"
3300 RETURN
3310 PRINT "KRINNG DESTROYED"
3320 SC(S3,S4)=0
3330 GL(Q1,Q2)=GL(Q1,Q2)-100
3340 KS=KS-1
3350 KL=KL-1
3360 IF KL=0 THEN 11100
3370 FOR A=1 TO 3
3380 IF (K(A,1)<>S3)+(K(A,2)<>S4)THEN 3430
3390 K(A,1)=0
3400 K(A,2)=0
3410 K(A,3)=0
3420 RETURN
3430 NEXT A
3440 RETURN
3450 PRINT "STARBASE DESTROYED"
3460 BA=0
3470 BL=BL-1
3480 SC(S3,S4)=0
3490 GL(Q1,Q2)=GL(Q1,Q2)-10
3500 RETURN
3510 PRINT "CANNOT DESTROY STARS"
3520 RETURN
3530 NEXT WP
3540 RETURN
3550 FOR WP=0 TO 7
3560 S3=INT(S1-WP*SIN(CS*CX)+.5)
3570 S4=INT(S2+WP*COS(CS*CX)+.5)
3580 IF (S3>7)+(S4>7)+(S3<1)+(S4<1)THEN 3540 ELSE 3260
3590 NEXT WP
4000 PRINT D$(4)
```

Listing 2-5—cont. Space War 7 Program

```
4010 IF KS>0 THEN 4040
4020 PRINT "SR SENSORS-NO KRINNG"
4030 RETURN
4040 IF D(7)>=0 THEN 4060
4050 PRINT "COMPUTER FAILURE HAMPERS ACCURACY"
4060 PRINT "PHASERS LOCKED":"ENERGY:";P2(EN)
4070 INPUT "UNITS TO FIRE->":X
4080 IF EN-X>0 THEN 4110
4090 PRINT "NOT ENOUGH ENERGY"
4100 RETURN
4110 TU=TU+.05
4120 IF TU>SD THEN 11050
4130 EN=EN-X
4140 IF C$="D" THEN 4160
4150 GOSUB 10000
4160 IF D(7)>=0 THEN 4180
4170 X=X*RND
4180 FOR I=1 TO 3
4190 IF K(I,3)<=0 THEN 4330
4200 H=(X/DT)+SGN(RND-.5)*7*RND
4210 K(I,3)=K(I,3)-H
4220 PRINT P2(H);" HIT ON KRINNG(";STR$(K(I,1));",";
     STR$(K(I,2));")":"LEAVES ";P2(K(I,3))
4230 IF K(I,3)>0 THEN 4330
4240 PRINT "KRINNG (";STR$(K(I,1));",";STR$(K(I,2));
     ")";" DESTROYED"
4250 KS=KS-1
4260 KL=KL-1
4270 IF KL=0 THEN 11100
4280 GL(Q1,Q2)=GL(Q1,Q2)-100
4290 SC(K(I,1),K(I,2))=0
4300 K(I,1)=0
4310 K(I,2)=0
4320 K(I,3)=0
4330 NEXT I
4340 RETURN
5000 PRINT "SHIELD ENERGY:";EN+SH:"UNITS TO SHIELDS->";
5010 INPUT X
5020 IF X<0 THEN 5060
5030 IF EN+SH-X<0 THEN 5000
5040 EN=EN+SH-X
5050 SH=X
5060 RETURN
6000 PRINT D$(6)
6010 GOSUB 14000
6020 INPUT "WARP FACTOR->":WP
6030 IF (WP<=0)THEN 6870
6040 IF (D(6)>=0)+(WP<=MAXSP)THEN 6070
6050 PRINT "ENGINES DAMAGED":"MAX SPEED:";MAXSP
6060 GOTO 6010
6070 TC=TC+1
6080 C$=""
6090 IF KS<=0 THEN 6110
6100 GOSUB 10000
6110 IF EN>12*WP THEN 6150
6120 IF SH<1 THEN 12000
6130 PRINT "ONLY ";P2(EN);" UNITS LEFT":
     "MAX WARP:";INT(EN/12)
```

Listing 2-5—cont. Space War 7 Program

```
6140 GOTO 6020
6150 FOR I=1 TO 9
6160 IF D(I)>=0 THEN 6210
6170 D(I)=D(I)+1
6180 IF D(I)<0 THEN 6210
6190 IF D(8)<0 THEN 6210
6200 PRINT D$(I);" REPAIRED"
6210 NEXT I
6220 IF RND>.1 THEN 6280
6230 R1=RD
6240 IF D(R1)>=0 THEN 6280
6250 IF D(8)<0 THEN 6280
6260 D(R1)=0
6270 PRINT :D$(R1);" REPAIRED"
6280 A1=7*Q2+S2-8
6290 B1=56-7*Q1-S1
6300 IF WP>=1 THEN 6320
6310 WP=WP*1.42858
6320 WP=WP*7
6330 EN=EN-2*WP
6340 TU=TU+WP/25
6350 IF TU>SD THEN 11050
6360 A2=DTA(WP)
6370 B2=DTB(WP)
6380 IF (A2<0)+(S2>48)+(B2<0)+(B2>48) THEN 6390
     ELSE 6420
6390 PRINT "STAY IN GALAXY"
6400 TU=TU+WP/24
6410 RETURN
6420 FOR X=0 TO INT(WP)
6430 IF (Q1=7-INT(DTB(X)/7))*(Q2=INT(DTA(X)/7+1))
     THEN 6460
6440 X=WP
6450 GOTO 6700
6460 S3=7-DTB(X)+7*INT(DTB(X)/7)
6470 S4=DTA(X)+1-7*INT(DTA(X)/7)
6480 IF SC(S3,S4)<2 THEN 6700
6490 SC(S1,S2)=0
6500 S1=7-DTB(X-1)+7*INT(DTB(X-1)/7)
6510 S2=DTA(X-1)+1-7*INT(DTA(X-1)/7)
6520 SC(S1,S2)=1
6530 PRINT "STOPPED AT (";STR$(S1);",";STR$(S2);")"
6540 TU=TU+WP/24
6550 PRINT "LOST ";P2(WP/24);"STARDATES"
6560 IF SC(S3,S4)<>4 THEN 6690
6570 PRINT "DOCKED."
6580 SH=2000
6590 TP=10
6600 C$="D"
6610 EN=3100-XA
6620 GOSUB 10000
6630 FOR I=1 TO 9
6640 IF D(I)>=0 THEN 6680
6650 D(I)=D(I)+(11*RND/RF)
6660 IF D(I)<0 THEN 6680
6670 D(I)=0
6680 NEXT I
6690 RETURN
```

Listing 2-5—cont. Space War 7 Program

```
6700 NEXT X
6710 SC(S1,S2)=0
6720 S1=7-B2+7*INT(B2/7)
6730 S2=A2+1-7*INT(A2/7)
6740 F1=7-INT(B2/7)
6750 F2=INT(A2/7)+1
6760 IF (Q1=F1)*(Q2=F2)THEN 6800
6770 Q1=F1
6780 Q2=F2
6790 GOSUB 13000
6800 SC(S1,S2)=1
6810 FOR A=S1-1 TO S1+1
6820 FOR B=S2-1 TO S2+1
6830 IF (A>7)+(B>7)+(A<1)+(B<1)THEN 6850
6840 IF SC(A,B)=4 THEN 6570
6850 NEXT B
6860 NEXT A
6870 RETURN
7000 INPUT "COMPUTER(1-6)->":A
7010 IF A=6 THEN 7070
7020 IF (A<6)*(A>0)THEN 7050
7030 PRINT :"COMMANDS:":"1=CHART":"2=STATUS":
     "3=TORPEDO COURSE":"4=POSITION":
     "5=DESTRUCT":"6=EXIT"
7040 GOTO 7000
7050 ON A GOSUB 7080,7220,7260,7540,7600
7060 IF A<>3 THEN 7000
7070 RETURN
7080 CALL CLEAR
7090 PRINT "GALAXY"
7100 FOR A=1 TO 7
7110 FOR B=1 TO 7
7120 IF GL(A,B)<>INT(GL(A,B))THEN 7150
7130 PRINT " ...";
7140 GOTO 7170
7150 ST$="..."&STR$(INT(GL(A,B)))
7160 PRINT " ";SEG$(ST$,LEN(ST$)-2,3);
7170 NEXT B
7180 PRINT : :
7190 NEXT A
7200 GOSUB 7540
7210 RETURN
7220 PRINT "STATUS":"KRINNG=";KL:"STARDATES=";
     P2(SD-TU):"BASES=";BL
7230 IF D(7)>=0 THEN 7250
7240 PRINT "NO ";D$(7)
7250 RETURN
7260 PRINT "COURSE":
7270 IF KS>0 THEN 7300
7280 PRINT "NO KRINNG"
7290 RETURN
7300 IF D(3)>=0 THEN 7330
7310 PRINT D$(3);" OUT"
7320 RETURN
7330 PRINT "COORD";" COURSE";"  DIST"
7340 FOR I=1 TO 3
7350 IF (K(I,3)<=0)THEN 7520
7360 F1=S1-K(I,1)
```

Listing 2-5—cont. Space War 7 Program

```
7370 F2=S2-K(I,2)
7380 WP=SQR(F1*F1+F2*F2)
7390 IF F1<=0 THEN 7420
7400 CS=ATN(F2/F1)
7410 GOTO 7460
7420 IF F1=0 THEN 7450
7430 CS=SGN(F2)*(3.14159-ATN(ABS(F2/F1)))
7440 GOTO 7460
7450 CS=SGN(F2)*3.14159/2
7460 CS=12-CS*1.9098593
7470 IF (F2<>0)+(F1>=0)THEN 7490
7480 CS=6
7490 IF CS<=12 THEN 7510
7500 CS=CS-12
7510 PRINT "(";STR$(K(I,1));",";STR$(K(I,2));")";
     TAB(7);P2(CS);TAB(14);P2(WP)
7520 NEXT I
7530 RETURN
7540 PRINT "QUADRANT (";STR$(Q1);",";STR$(Q2);")":
     "SECTOR (";STR$(S1);",";STR$(S2);")"
7550 IF SH>200*KS THEN 7590
7560 PRINT "SHIELD ENERGY LOW"
7570 IF D(6)>=0 THEN 7590
7580 PRINT "NO ";D$(6)
7590 RETURN
7600 INPUT "ENTER CODE ":ST$
7610 IF ST$=SC$ THEN 7640
7620 PRINT "ABORTED"
7630 RETURN
7640 PRINT "COUNTDOWN BEGUN":"5":"4":"3":"2":"1"
     : :"DESTROYED!"
7650 STOP
8000 PRINT "DAMAGE REPORT":"DEVICE ";TAB(20);"STATE "
8010 FOR I=1 TO 9
8020 PRINT D$(I);TAB(20);P2(D(I))
8030 NEXT I
8040 RETURN
9000 PRINT "DAMAGE REPAIR"
9010 AV=INT(RK/2)
9020 IF TC>=AV THEN 9050
9030 PRINT "REPAIR INACTIVE":"NEEDS";AV-TC;"MORE UNITS"
9040 RETURN
9050 IF C$="D" THEN 9080
9060 PRINT :"MUST BE DOCKED FOR REPAIRS"
9070 RETURN
9080 DM=0
9090 FOR A=1 TO 9
9100 IF D(A)>=0 THEN 9130
9110 DM=1
9120 PRINT A;" ";D$(A);" DAMAGED"
9130 NEXT A
9140 IF DM=0 THEN 9240
9150 INPUT "REPAIR WHICH? ":A
9160 IF (A<1)+(A>9)THEN 9150
9170 IF D(A)<0 THEN 9200
9180 PRINT D$(A);" OK"
9190 GOTO 9150
9200 PRINT D$(A);" WORKING"
```

Listing 2-5—cont. Space War 7 Program

```
9210 D(A)=0
9220 TC=TC-1
9230 RETURN
9240 PRINT "NO DAMAGE"
9250 RETURN
10000 IF KS=0 THEN 10040
10010 PRINT "KRINNG HITS SHIP"
10020 IF C$<>"D" THEN 10050
10030 PRINT "BASE PROTECTS SHIP"
10040 RETURN
10050 FOR I=1 TO 3
10060 IF K(I,3)<=0 THEN 10200
10070 H=K(I,3)/DT+SGN(RND-.5)*RND*7
10080 SH=SH-H
10090 PRINT P2(H);" HIT FROM (";STR$(K(I,1));",";
      STR$(K(I,2));")":"LEAVES ";P2(SH)
10100 IF SH<0 THEN 11070
10110 GOSUB 10220
10120 IF RND>RF/15 THEN 10200
10130 R1=RD
10140 R2=RD
10150 IF SC(R1,R2)<>0 THEN 10130
10160 SC(R1,R2)=3
10170 SC(K(I,1),K(I,2))=0
10180 K(I,1)=R1
10190 K(I,2)=R2
10200 NEXT I
10210 RETURN
10220 IF RND>RF/23 THEN 10280
10230 R1=INT(RND*9+1)
10240 IF D(R1)<0 THEN 10280
10250 D(R1)=D(R1)-10*RND-1
10260 IF D(8)<0 THEN 10280
10270 PRINT D$(R1);" OUT"
10280 RETURN
11000 PRINT :"ENTERPRISE DEAD IN SPACE"
11010 GOTO 11080
11020 IF KS<=0 THEN 11080
11030 GOSUB 10000
11040 GOTO 11020
11050 PRINT "IT IS STARDATE ";P2(TU)
11060 GOTO 11080
11070 PRINT "ENTERPRISE DISABLED"
11080 PRINT KL;" KRINNG REMAIN": :"YOU HAVE FAILED!"
11090 GOTO 11110
11100 PRINT "FEDERATION IS SAVED"
11110 L=(U-KL)*10+((U-KL)*500/TU)-100*(Y-BL)
11120 IF (EN+SH)>0 THEN 11140
11130 L=L-200
11140 IF KL<>0 THEN 11160
11150 L=L+(RF*100)
11160 PRINT "RATING:";P2(L)
11170 STOP
12000 INPUT "ENTER SECURITY CODE:":SC$
12010 INPUT "ENTER RANK(1-12):":RF
12020 I=RF*20
12030 RK=RF
12040 RF=RF+.0001
```

Listing 2-5—cont. Space War 7 Program

```
12050 PRINT "RISK RATE ";I: :"SETTING UP GALAXY"
12060 CX=.0174532925
12070 XA=INT(RK*50)
12080 MAXSP=P2(10/(RK+.1))
12090 EN=3100-XA
12100 SH=2000
12110 TP=10
12120 Q1=RD
12130 Q2=RD
12140 S1=RD
12150 S2=RD
12160 FOR I=1 TO 9
12170 READ D$(I)
12180 NEXT I
12190 A$(1)="."
12200 A$(2)="E"
12210 A$(3)="*"
12220 A$(4)="K"
12230 A$(5)="B"
12240 F1=.86
12250 F2=.01
12260 F3=.95
12270 F4=.99
12280 KL=0
12290 BL=0
12300 FOR I=1 TO 7
12310 FOR J=1 TO 7
12320 R1=RND
12330 R2=RND
12340 KS=-(R1>F1-F2*RF)-(R1>F3-F2*RF)-(R1>F4-F2*RF)
12350 KL=KL+KS
12360 BA=-(R2>F3)
12370 BL=BL+BA
12380 GL(I,J)=100*KS+10*BA+RD
12390 IF BA=0 THEN 12410
12400 GL(I,J)=GL(I,J)+.5
12410 NEXT J
12420 NEXT I
12430 IF (BL<1)+(KL<1)THEN 12280
12440 PRINT "KRINNG ";KL;" BASES ";BL
12450 U=KL
12460 Y=BL
12470 SD=KL+RND*KL/RF+10-RF
12480 PRINT "STARDATES ";P2(SD)
13000 S3=0
13010 BA=S3
13020 KS=BA
13030 FOR I=1 TO 7
13040 FOR J=1 TO 7
13050 SC(I,J)=0
13060 NEXT J
13070 NEXT I
13080 FOR I=1 TO 3
13090 FOR J=1 TO 3
13100 K(I,J)=0
13110 NEXT J
13120 NEXT I
13130 SC(S1,S2)=1
```

Listing 2-5—cont. Space War 7 Program

```
13140 X=.01*GL(Q1,Q2)
13150 KS=INT(X)
13160 Y=(X-KS)*10
13170 BA=INT(Y)
13180 S3=GL(Q1,Q2)-100*KS-10*BA
13190 IF KS<>0 THEN 13260
13200 FOR I=1 TO 3
13210 FOR J=1 TO 3
13220 K(I,J)=0
13230 NEXT J
13240 NEXT I
13250 GOTO 13360
13260 PRINT "CONDITION RED!!!!"
13270 FOR I=1 TO KS
13280 R1=RD
13290 R2=RD
13300 IF SC(R1,R2)<>0 THEN 13280
13310 SC(R1,R2)=3
13320 K(I,1)=R1
13330 K(I,2)=R2
13340 K(I,3)=200
13350 NEXT I
13360 IF BA=0 THEN 13410
13370 R1=RD
13380 R2=RD
13390 IF SC(R1,R2)<>0 THEN 13370
13400 SC(R1,R2)=4
13410 IF S3=0 THEN 13480
13420 FOR I=1 TO S3
13430 R1=RD
13440 R2=RD
13450 IF SC(R1,R2)<>0 THEN 13430
13460 SC(R1,R2)=2
13470 NEXT I
13480 GL(Q1,Q2)=INT(GL(Q1,Q2))+.5
13490 GOSUB 7540
13500 RETURN
14000 INPUT "DIRECTION(0-12)->":CS
14010 IF (CS<0)+(CS>12)THEN 14000
14020 IF CS>0 THEN 14040
14030 CS=12
14040 CS=(12-CS)*30+90
14050 IF CS<360 THEN 14070
14060 CS=CS-360
14070 RETURN
25000 DATA SHORT RANGE SCAN,LONG RANGE SCAN
25010 DATA TORPEDOES,PHASERS,SHIELDS,WARP DRIVE
25020 DATA COMPUTER,DAMAGE REPORT,DAMAGE CONTROL
```

TIC-TAC-TOE PROGRAM

What it does:

Tic-Tac-Toe is an amusing game, especially for children. The Tic-Tac-Toe game program makes liberal use of the graphics capability of the TI-99/4A.

It's very simple to play this game. Tell the computer whether there will be two human players or whether you prefer to play against it. If you choose to play against the computer, it asks if you would like to go first.

Each square on the Tic-Tac-Toe grid has a number printed in it. Use these numbers, ranging from one (1) to nine (9), to tell the program which square you want to play.

What it shows you:

This Tic-Tac-Toe program shows two items that may be of interest to anyone interested in programming. First are the graphics character definitions (statements 4090–4140). These CALL CHAR statements define the shapes used to construct the playing grid and the two play tokens (the heart and the bow).

These special characters were designed using the Character Editor program also included in this book.

The grid definition characters are combined to form the row pattern array (ROWPAT$) used to print the grid to the screen.

Another interesting aspect of this program is the direct positioning input/output routines in the 20000 and 21000 statement number range. These routines simulate some of the behavior of the DISPLAY AT and ACCEPT FROM Extended BASIC statements.

What you can do:

It's relatively easy to change the color combinations we've chosen for the board and play grid. If you prefer a different set of colors, by all means, make things the colors that you like.

You may also prefer shapes for the play tokens other than the heart and bow we provide. Use the Character Editor program to design your own characters and experiment with character/background color combinations until you find the one you like best.

Table 2-11. Tic-Tac-Toe Routines

Lines	Function
100-300	Initialize variables and screen.
310-490	Main game control loop.
1000-1230	Get player's move.
2000-2120	Sum win vectors; check for winner.
3000-3060	Print the play grid to the screen.
4000-4360	Initialize colors and character patterns.
5000-5070	Place a move on the screen grid.
6000-6900	Computer move logic.
8000-8180	Computer search for winning or blocking move.
9000-9120	Dialog for one or two player game.
20000-20090	Direct print of a message to the screen.
21000-21150	Direct read of a number from the keyboard.

Table 2-12. Tic-Tac-Toe Variables

Variable	Description
BOARD(9)	Board squares (0 = not played; 1 = player one; −1 = player two).
P$(2)	Player token characters.
ROWPAT$(11)	Playing grid row patterns.
SUM(8)	Sum of the winning vectors.
VECT(8,3)	Winning vectors (loaded from DATA statements at lines 4330–4350).

Listing 2-6. Tic-Tac-Toe Program

```
100 REM  PLAY TIC TAC TOE
110 REM  USE GRAPHICS CHARACTERS FOR PIECES
120 REM
130 CALL CLEAR
140 REM ** INITIALIZE GAME
150 DIM ROWPAT$(11)
160 REM P$ ARRAY SHOWS PLAYER TOKENS
170 P$(1)=CHR$(128)
180 P$(2)=CHR$(136)
190 GOSUB 4000
200 REM
210 REM ** INITIALIZE BOARD
220 FOR I=1 TO 9
230 BOARD(I)=0
240 NEXT I
250 FOR I=1 TO 8
260 SUM(I)=0
270 NEXT I
280 GOSUB 8000
290 REM
300 GOSUB 3000
310 FOR ROUND=1 TO 9
320 MOVE=0
330 PLAYER=3-PLAYER
```

Listing 2-6—cont. Tic-Tac-Toe Program

```
340 IF (NUMPLAYERS=2)+(PLAYER=2)THEN 390
350 GOSUB 7000
360 IF MOVE<>0 THEN 400
370 GOSUB 6000
380 GOTO 400
390 GOSUB 1000
400 GOSUB 5000
410 GOSUB 2000
420 IF WIN<>0 THEN 460
430 NEXT ROUND
440 PRINT "THE GAME IS A DRAW"
450 GOTO 470
460 PRINT :"GOOD PLAYING "&P$(PLAYER)
470 INPUT "WANT TO PLAY AGAIN (Y/N) ":Y$
480 IF Y$="Y" THEN 210
490 STOP
1000 REM GET MOVE
1010 MSG$="YOUR MOVE(1-9) "&P$(PLAYER)
1020 MCOL=3
1030 MROW=22
1040 GOSUB 9000
1050 MCOL=MCOL+1
1060 GOSUB 10000
1070 MOVE=ANS
1080 IF (MOVE<1)+(MOVE>9)THEN 1120
1090 IF BOARD(MOVE)<>0 THEN 1220
1100 BOARD(MOVE)=(2*PLAYER)-3
1110 RETURN
1120 MSG$="MOVES MUST BE 1-9"
1130 MROW=23
1140 MCOL=3
1145 CALL SOUND(-300,131,0,262,2,-2,1)
1150 GOSUB 9000
1160 FOR I=1 TO 250
1170 NEXT I
1180 MSG$=""
1190 OCOL=3
1200 GOSUB 9000
1210 GOTO 1010
1220 MSG$="SQUARE ALREADY PLAYED!!"
1230 GOTO 1130
2000 REM  CHECK FOR WIN
2010 REM IF BOARD=PLAYER THEN PLAYER WINS
2020 WIN=PLAYER
2030 SUM(1)=BOARD(1)+BOARD(2)+BOARD(3)
2040 SUM(2)=BOARD(4)+BOARD(5)+BOARD(6)
2050 SUM(3)=BOARD(7)+BOARD(8)+BOARD(9)
2060 SUM(4)=BOARD(1)+BOARD(4)+BOARD(7)
2070 SUM(5)=BOARD(2)+BOARD(5)+BOARD(8)
2080 SUM(6)=BOARD(3)+BOARD(6)+BOARD(9)
2090 SUM(7)=BOARD(1)+BOARD(5)+BOARD(9)
2100 SUM(8)=BOARD(3)+BOARD(5)+BOARD(7)
2110 FOR I=1 TO 8
2120 IF ABS(SUM(I))=3 THEN 2160
2130 NEXT I
2140 REM NO WINNER
2150 WIN=0
2160 RETURN
```

Listing 2-6—cont. Tic-Tac-Toe Program

```
3000 REM WRITE BOARD
3010 CALL CLEAR
3020 FOR ROW=1 TO 11
3030 PRINT TAB(8);ROWPAT$(ROW)
3040 NEXT ROW
3050 PRINT : : : : :
3060 RETURN
4000 REM INITIALIZE
4010 CALL SCREEN(13)
4020 FOR I=1 TO 14
4030 CALL COLOR(I,16,1)
4040 NEXT I
4050 CALL COLOR(3,2,16)
4060 CALL COLOR(4,2,16)
4070 CALL COLOR(13,7,16)
4080 CALL COLOR(14,5,16)
4090 CALL CHAR(59,"0000000000000000")
4100 CALL CHAR(60,"1818181818181818")
4110 CALL CHAR(61,"181818FFFF181818")
4120 CALL CHAR(62,"000000FFFF000000")
4130 CALL CHAR(128,"66FFFFFF7E3C1818")
4140 CALL CHAR(136,"81C3E7FFFFE7C381")
4150 ROWPAT$(1)="1"&CHR$(59)&CHR$(59)&CHR$(60)&"2"&
     CHR$(59)&CHR$(59)&CHR$(60)&"3"&CHR$(59)&CHR$(59)
4160 ROWPAT$(5)="4"&CHR$(59)&CHR$(59)&CHR$(60)&"5"&
     CHR$(59)&CHR$(59)&CHR$(60)&"6"&CHR$(59)&CHR$(59)
4170 ROWPAT$(9)="7"&CHR$(59)&CHR$(59)&CHR$(60)&"8"&
     CHR$(59)&CHR$(59)&CHR$(60)&"9"&CHR$(59)&CHR$(59)
4180 ROWA$=CHR$(59)&CHR$(59)&CHR$(59)&CHR$(60)&
     CHR$(59)&CHR$(59)&CHR$(59)&CHR$(60)&CHR$(59)&
     CHR$(59)&CHR$(59)
4190 ROWB$=CHR$(62)&CHR$(62)&CHR$(62)&CHR$(61)&
     CHR$(62)&CHR$(62)&CHR$(62)&CHR$(61)&CHR$(62)&
     CHR$(62)&CHR$(62)
4200 ROWPAT$(2)=ROWA$
4210 ROWPAT$(3)=ROWA$
4220 ROWPAT$(4)=ROWB$
4230 ROWPAT$(6)=ROWA$
4240 ROWPAT$(7)=ROWA$
4250 ROWPAT$(8)=ROWB$
4260 ROWPAT$(10)=ROWA$
4270 ROWPAT$(11)=ROWA$
4280 FOR I=1 TO 8
4290 FOR J=1 TO 3
4300 READ VECT(I,J)
4310 NEXT J
4320 NEXT I
4330 DATA 1,2,3,4,5,6,7,8,9
4340 DATA 1,4,7,2,5,8,3,6,9
4350 DATA 1,5,9,3,5,7
4360 RETURN
5000 REM SHOW THE MOVE
5010 ROW=4*(INT(MOVE/3.5)+1)+5
5020 FOR COL=MOVE TO 1 STEP -3
5030 IF COL<4 THEN 5050
5040 NEXT COL
5050 COL=11+((COL-1)*4)
5060 CALL HCHAR(ROW,COL,ASC(P$(PLAYER)))
```

Listing 2-6—cont. Tic-Tac-Toe Program

```
5070 RETURN
6000 REM MAKE THE COMPUTER
6010 REM  MOVE
6020 IF ROUND>4 THEN 6730
6030 ON ROUND GOTO 6040,6070,6180,6250
6040 REM FIRST MOVE GO FOR 1
6050 MOVE=1
6060 GOTO 6710
6070 REM SECOND MOVE
6080 MOVE=5
6090 IF (BOARD(1)=1)+(BOARD(3)=1)+(BOARD(7)=1)+
     (BOARD(9)=1)THEN 6710
6100 MOVE=1
6110 IF (SUM(1)=1)+(SUM(4)=1)THEN 6710
6120 MOVE=9
6130 IF (SUM(3)=1)+(SUM(6)=1)THEN 6710
6140 MOVE=3
6150 IF (SUM(1)=1)+(SUM(6)=1)THEN 6710
6160 MOVE=7
6170 GOTO 6710
6180 REM THIRD MOVE
6190 MOVE=5
6200 IF (BOARD(1)=1)+(BOARD(3)=1)+(BOARD(7)=1)+
     (BOARD(9)=1)THEN 6710
6210 MOVE=9
6220 IF BOARD(9)=0 THEN 6710
6230 MOVE=7
6240 GOTO 6710
6250 REM FOURTH MOVE
6260 MOVE=5
6270 IF BOARD(5)=0 THEN 6710
6280 IF BOARD(5)<>-1 THEN 6520
6290 MOVE=6
6300 IF (BOARD(1)=1)*(BOARD(9)=1)THEN 6710
6310 MOVE=4
6320 IF (BOARD(3)=1)*(BOARD(7)=1)THEN 6710
6330 IF BOARD(1)<>1 THEN 6380
6340 MOVE=3
6350 IF BOARD(6)=1 THEN 6710
6360 MOVE=7
6370 GOTO 6710
6380 IF BOARD(3)<>1 THEN 6430
6390 MOVE=9
6400 IF BOARD(8)=1 THEN 6710
6410 MOVE=1
6420 GOTO 6710
6430 IF BOARD(7)<>1 THEN 6480
6440 MOVE=1
6450 IF BOARD(2)=1 THEN 6710
6460 MOVE=9
6470 GOTO 6710
6480 MOVE=7
6490 IF BOARD(4)=1 THEN 6710
6500 MOVE=3
6510 GOTO 6710
6520 REM NO CENTER SPACE
6530 IF BOARD(1)<>-1 THEN 6580
6540 MOVE=9
```

Listing 2-6—cont. Tic-Tac-Toe Program

```
6550 IF BOARD(9)=0 THEN 6710
6560 MOVE=7
6570 GOTO 6710
6580 IF BOARD(3)<>-1 THEN 6630
6590 MOVE=7
6600 IF BOARD(7)=0 THEN 6710
6610 MOVE=9
6620 GOTO 6710
6630 IF BOARD(7)<>-1 THEN 6680
6640 MOVE=3
6650 IF BOARD(3)=0 THEN 6710
6660 MOVE=1
6670 GOTO 6710
6680 MOVE=1
6690 IF BOARD(1)=0 THEN 6710
6700 MOVE=3
6710 BOARD(MOVE)=-1
6720 RETURN
6730 REM LOOK FOR A MOVE
6740 MOVE=1
6750 IF BOARD(1)<>0 THEN 6770
6760 IF (SUM(1)=1)+(SUM(4)=1)THEN 6710
6770 MOVE=3
6780 IF BOARD(3)<>0 THEN 6800
6790 IF (SUM(1)=1)+(SUM(6)=1)THEN 6710
6800 MOVE=7
6810 IF BOARD(7)<>0 THEN 6830
6820 IF (SUM(3)=1)+(SUM(4)=1)THEN 6710
6830 MOVE=9
6840 IF BOARD(9)<>0 THEN 6860
6850 IF (SUM(3)=1)+(SUM(6)=1)THEN 6710
6860 FOR MOVE=1 TO 9
6870 IF BOARD(MOVE)<>0 THEN 6900
6880 BOARD(MOVE)=-1
6890 RETURN
6900 NEXT MOVE
7000 REM CHECK FOR WIN MOVE
7010 REM AND FOR BLOCKING MOVE
7020 FOR I=1 TO 8
7030 IF SUM(I)=-2 THEN 7090
7040 NEXT I
7050 FOR I=1 TO 8
7060 IF SUM(I)=2 THEN 7130
7070 NEXT I
7080 RETURN
7090 FOR J=1 TO 3
7100 MOVE=VECT(I,J)
7110 IF BOARD(MOVE)=0 THEN 7170
7120 NEXT J
7130 FOR J=1 TO 3
7140 MOVE=VECT(I,J)
7150 IF BOARD(MOVE)=0 THEN 7170
7160 NEXT J
7170 BOARD(MOVE)=-1
7180 RETURN
8000 REM SELECT ONE OR TWO PLAYER GAME
8010 NUMPLAYERS=2
8020 PLAYER=2
```

Listing 2-6—cont. Tic-Tac-Toe Program

```
8030 PRINT :"WILL THERE BE TWO"
8040 INPUT "  HUMAN PLAYERS? ":Y$
8050 IF Y$="Y" THEN 8120
8060 IF Y$<>"N" THEN 8030
8070 NUMPLAYERS=1
8080 INPUT "DO YOU WANT TO GO FIRST? ":Y$
8090 IF Y$="N" THEN 8120
8100 IF Y$<>"Y" THEN 8080
8110 PLAYER=1
8120 RETURN
9000 REM PRINT MSG$ AT MROW STARTING IN MCOL
9010 MLEN=LEN(MSG$)
9020 IF MLEN=0 THEN 9080
9030 FOR OCOL=1 TO MLEN
9040 CALL HCHAR(MROW,OCOL+MCOL,ASC(SEG$(MSG$,OCOL,1)))
9050 NEXT OCOL
9060 MCOL=MCOL+OCOL
9070 RETURN
9080 CALL HCHAR(MROW,OCOL,32,MCOL-OCOL+1)
9090 RETURN
10000 REM GET A NUMERIC ANSWER
10010 FOR OCOL=0 TO 1
10020 CALL HCHAR(MROW,MCOL+OCOL,32)
10030 NEXT OCOL
10040 DIGITS=0
10050 CALL SOUND(-150,1760,0)
10060 CALL KEY(0,ONE,S)
10070 IF S=0 THEN 10060
10080 CALL HCHAR(MROW,MCOL,ONE)
10090 ONE=ONE-48
10100 IF (ONE>=0)*(ONE<10)THEN 10140
10110 CALL SOUND(-300,131,0,262,2,-2,1)
10120 GOTO 10060
10130 DIGITS=1
10140 MCOL=MCOL+1
10150 ANS=ONE
10160 RETURN
```

WUMPUS PROGRAM

What it does:

Hunt the Wumpus is a more simple, child's form of adventure game. You wander through a maze of twenty caves in search of the mythical Wumpus. In your arsenal, you have five arrows. You must shoot the deadly Wumpus before he finds you. Because if he does, YOU'RE DEAD.

You and the Wumpus aren't the only things in the twenty caves. Some of the caves contain bottomless pits (well, almost bottomless). If you fall into one of those, too bad!

There are also bats who pick you up and then drop you at some random location in the maze of caves. If they drop you on the Wumpus, or into a pit, you're done for.

Hunt the Wumpus is a game very different from the arcade-type games. It demands some logical thinking and planning. You can beat the Wumpus, if you try!

What it shows you:

The Wumpus game uses several sound effects and some music. If you're interested in making sounds with your TI computer, these sound routines may be of interest to you.

What you can do:

You can increase the size of the maze. Note that it's easier to add two caves at a time to the existing maze.

If you like sound effects, try adding a sound for the Wumpus when it moves.

You can make the game more difficult, especially if you increase the size of the maze, by adding more bats and pits.

Table 2-13. Wumpus Routines

Lines	Function
100-560	Initialization and game instructions.
570-800	Main game control loop.
1000-1420	Game rules.
2000-2160	Check your location and print warnings.
3000-3070	Get shoot or move command.
4000-4470	Shoot an arrow and move the Wumpus.
5000-5260	You move and encounter hazards.
6000-6050	Initialize screen and character colors.
7000-7070	Arrow flight sound.
8000-8140	Fall into pit sound.
9000-9090	Bat snatch sound.
10000-10100	You win music.
11000-11180	You lose music.
12000-12040	Delay routine.
13000-13040	Cave layout data.

Table 2-14. Wumpus Variables

Variable	Description
ARROWS	The number of arrows you have left.
CAVE(20,3)	Holds the room numbers of the three rooms connected to each room in the cave.
HOLDMAZE(6)	Stores the data in LOCS so that the same game can be replayed.
LOCS(6)	Holds the room numbers of you, the Wumpus, the bats, and the pits.
MAXROOMS	The maximum number of rooms in the maze. Now set to twenty.
NOTE(20)	Array to hold the musical notes.
PATH(5)	The path of your arrow.
ROOMS	Number of rooms to shoot your arrow.
SHORMOV	Flag indicates whether to shoot (1) or move (2).
TIME(4)	Note times for the music.
WIN	Set to: zero (0) when there is no winner; one (1) when you win; and minus one (−1) when the Wumpus wins.
USER FUNCTION	
RNDX(X)	Produces a random integer based on argument X.

Listing 2-7. Wumpus Program

```
100 CALL CLEAR
110 GOSUB 6000
120 RANDOMIZE
130 PRINT TAB(5);"**** WUMPUS ****": : : : : :
140 FOR I=1 TO 75
150 NEXT I
160 DIM PATH(5),CAVE(20,3),LOCS(6),HOLDMAZE(6),NOTE(20)
170 DEF RNDX(X)=INT(X*RND)+1
180 MAXROOMS=20
190 PRINT "Do you want instructions"
200 INPUT "  (Y/N)->":ANS$
210 IF (SEG$(ANS$,1,1)="N")+(SEG$(ANS$,1,1)="n")
    THEN 320
220 IF (SEG$(ANS$,1,1)="Y")+(SEG$(ANS$,1,1)="y")
    THEN 250
230 PRINT : :"Yes or no, please.": :
240 GOTO 190
250 GOSUB 1000
260 PRINT "Do you understand the"
270 INPUT " instructions (Y/N)->":ANS$
280 IF (SEG$(ANS$,1,1)="N")+(SEG$(ANS$,1,1)="n")
    THEN 250
290 IF (SEG$(ANS$,1,1)="Y")+(SEG$(ANS$,1,1)="y")
    THEN 320
300 PRINT : :"Yes or no, please.": :
310 GOTO 260
320 REM FILL THE CAVE
330 CALL CLEAR
340 RESTORE 13010
350 FOR J=1 TO MAXROOMS
360 FOR K=1 TO 3
370 READ CAVE(J,K)
```

Listing 2-7—cont. Wumpus Program

```
380 NEXT K
390 NEXT J
400 REM LOCATIONS:
410 REM  1 IS YOU
420 REM  2 IS WUMPUS
430 REM  3&4 ARE PITS
440 REM  5&6 ARE BATS
450 FOR J=1 TO 6
460 LOCS(J)=RNDX(MAXROOMS)
470 HOLDMAZE(J)=LOCS(J)
480 REM CHECK FOR DUPLICATES
490 FOR K=1 TO J-1
500 IF LOCS(J)=LOCS(K)THEN 460
510 NEXT K
520 NEXT J
530 ARROWS=5
540 LOCTMP=LOCS(1)
550 CALL CLEAR
560 PRINT : : :TAB(7);"HUNT THE WUMPUS": : :
570 GOSUB 2000
580 GOSUB 3000
590 ON SHORMOV GOSUB 4000,5000
600 IF WIN=0 THEN 570
610 IF WIN>0 THEN 650
620 GOSUB 11000
630 PRINT : : :"HA HA HA!  YOU LOSE!!": : :
640 GOTO 670
650 GOSUB 10000
660 PRINT : : :"YOU WIN!!!!": :"The Wumpus'll getcha":
    "   next time!!": : :
670 INPUT "Want to play again (Y/N)->":ANS$
680 IF (SEG$(ANS$,1,1)="N")+(SEG$(ANS$,1,1)="n")
    THEN 800
690 IF (SEG$(ANS$,1,1)<>"Y")*(SEG$(ANS$,1,1)<>"y")
    THEN 670
700 PRINT : : :
710 FOR J=1 TO 6
720 LOCS(J)=HOLDMAZE(J)
730 NEXT J
740 INPUT "Same set up (Y/N)-> ":ANS$
750 CALL SCREEN(13)
760 IF (SEG$(ANS$,1,1)="N")+(SEG$(ANS$,1,1)="n")
    THEN 450
770 IF (SEG$(ANS$,1,1)="Y")+(SEG$(ANS$,1,1)="y")
    THEN 530
780 PRINT : :"Yes or no, please.": :
790 GOTO 740
800 STOP
1000 REM INSTRUCTIONS
1010 CALL CLEAR
1020 PRINT "Welcome to 'HUNT THE WUMPUS'": :
     "  The WUMPUS lives in a 20"
1030 PRINT "room cave.  Each room has":
     "three tunnels leading to"
1040 PRINT "other rooms.": :"  There are HAZARDS!"
1050 PRINT :"BOTTOMLESS PITS - two rooms":
     "contain bottomless pits. If"
```

Listing 2-7—cont. Wumpus Program

```
1060 PRINT "you go there you fall into":
     "the pit -- and die!!"
1070 PRINT :"BIG BATS - two rooms harbor":
     "huge bats who snatch you and"
1080 PRINT "drop you in another room":
     "(where you could be in BIG":"TROUBLE)."
1090 GOSUB 12000
1100 CALL CLEAR
1110 PRINT "  The WUMPUS!!": : :
     "  The WUMPUS is immune to"
1120 PRINT "the hazards (he has sucker":
     "feet and is too big for a"
1130 PRINT "bat to lift).": :"  He's usually asleep."
1140 PRINT :"  BUT, two things wake him"
1150 PRINT "up: when you enter his room":
     "and when you shoot an arrow."
1160 PRINT :"  When the WUMPUS wakes, he"
1170 PRINT "often moves to another room.":
     "If he moves to the room"
1180 PRINT "you're in, he EATS you!!":::
1190 GOSUB 12000
1200 CALL CLEAR
1210 PRINT "  RULES:": :"Each turn, you may MOVE, or"
1220 PRINT "you may SHOOT a crooked":"arrow."
1230 PRINT :"MOVING: You can move one":"room."
1240 PRINT :"ARROWS: You have 5 arrows.":
     "You lose when you run out."
1250 PRINT :"  Your arrows are crooked so":
     "they will follow the"
1260 PRINT "tunnels.  Your bow shoots an":
     "arrow through up to 5 rooms."
1270 PRINT "You aim by telling the":
     "computer which rooms you"
1280 PRINT "want the arrow to go":"through."
1290 GOSUB 12000
1300 CALL CLEAR
1310 PRINT "  If the arrow can't go the":
     "way you said, it moves at"
1320 PRINT "random to the next room.":
     "  If the arrow hits the"
1330 PRINT "WUMPUS, you WIN!!!!":
     "  If the arrow hits you, you":"LOSE!!!"
1340 PRINT :"  SOME WARNINGS!!!"
1350 PRINT :"  When you are one room away":
     "from the WUMPUS or a HAZARD,"
1360 PRINT "the computer warns you:"
1370 PRINT :"WUMPUS - 'I smell a WUMPUS'"
1380 PRINT :"BAT  -  'BATS nearby'"
1390 PRINT :"PIT  -  'I feel a draft'"
1400 GOSUB 12000
1410 CALL CLEAR
1420 RETURN
2000 REM SEE IF YOU ARE ANYWHERE THAT MATTERS
2010 PRINT
2020 FOR J=2 TO 6
2030 FOR K=1 TO 3
2040 IF CAVE(LOCS(1),K)<>LOCS(J)THEN 2110
```

Listing 2-7—cont. Wumpus Program

```
2050 ON J-1 GOTO 2060,2080,2080,2100,2100
2060 PRINT : :"I smell a WUMPUS!!": :
2070 GOTO 2100
2080 PRINT : :"I feel a draft.": :
2090 GOTO 2110
2100 PRINT : :"BATS nearby!!": :
2110 NEXT K
2120 NEXT J
2130 PRINT : :"You are in room ";LOCTMP
2140 PRINT : :"Tunnels lead to ";CAVE(LOCTMP,1);
     ",";CAVE(LOCTMP,2);
2150 PRINT "&";CAVE(LOCTMP,3): :
2160 RETURN
3000 REM GET MOVE
3010 INPUT "Shoot(S) or Move(M)->":ANS$
3020 IF (SEG$(ANS$,1,1)<>"S")*(SEG$(ANS$,1,1)<>"s")
     THEN 3050
3030 SHORMOV=1
3040 RETURN
3050 IF (SEG$(ANS$,1,1)<>"M")*(SEG$(ANS$,1,1)<>"m")
     THEN 3010
3060 SHORMOV=2
3070 RETURN
4000 REM SHOOT ARROW
4010 WIN=0
4020 INPUT "How many rooms(1-5)->":ROOMS
4030 IF (ROOMS<1)+(ROOMS>5)+(ROOMS<>INT(ROOMS))
     THEN 4020
4040 FOR K=1 TO ROOMS
4050 INPUT "Room # ":PATH(K)
4060 IF K<=2 THEN 4100
4070 IF PATH(K)<>PATH(K-2)THEN 4100
4080 PRINT : :"Arrows aren't that crooked.":
     " Try another room.": :
4090 GOTO 4050
4100 NEXT K
4110 REM ARROW SOUND
4120 GOSUB 7000
4130 LOCTMP=LOCS(1)
4140 FOR K=1 TO ROOMS
4150 FOR K1=1 TO 3
4160 IF CAVE(LOCTMP,K1)=PATH(K)THEN 4330
4170 NEXT K1
4180 REM RANDOM ARROW PATH
4190 LOCTMP=CAVE(LOCTMP,RNDX(3))
4200 GOTO 4340
4210 NEXT K
4220 PRINT : :"Missed him!!": :
4230 LOCTMP=LOCS(1)
4240 REM GO MOVE WUMPUS
4250 GOSUB 4410
4260 REM YOU SHOT AN ARROW
4270 ARROWS=ARROWS-1
4280 IF ARROWS>0 THEN 4310
4290 PRINT : : :"OH NO!!":
     "You've run out of arrows!!": : :
4300 WIN=-1
```

Listing 2-7—cont. Wumpus Program

```
4310 RETURN
4320 REM CHECK FOR HIT
4330 LOCTMP=PATH(K)
4340 IF LOCTMP<>LOCS(2)THEN 4380
4350 PRINT : : :"AHA!!  YOU GOT THE WUMPUS!!!": : :
4360 WIN=1
4370 RETURN
4380 IF LOCTMP<>LOCS(1)THEN 4210
4390 PRINT : : :"OUCH!! The arrow got YOU!!!": : :
4400 GOTO 4300
4410 REM WUMPUS MOVES
4420 K=RNDX(4)
4430 IF K=4 THEN 4450
4440 LOCS(2)=CAVE(LOCS(2),K)
4450 IF LOCS(2)<>LOCTMP THEN 4480
4460 PRINT : : :"Too bad!!!!": :
     "The WUMPUS got YOU!!!": : :
4470 WIN=-1
4480 RETURN
5000 REM YOU MOVE
5010 WIN=0
5020 INPUT "Where to->":LOCTMP
5030 IF (LOCTMP<1)+(LOCTMP>MAXROOMS)+
     (LOCTMP<>INT(LOCTMP))THEN 5020
5040 FOR K=1 TO 3
5050 IF CAVE(LOCS(1),K)=LOCTMP THEN 5100
5060 NEXT K
5070 IF LOCTMP=LOCS(1)THEN 5100
5080 PRINT : :"Can't get there from here.": : :
5090 GOTO 5020
5100 LOCS(1)=LOCTMP
5110 IF LOCTMP<>LOCS(2)THEN 5140
5120 PRINT : : :"... OOPS! Bumped a WUMPUS!!"
5130 GOTO 4450
5140 IF (LOCTMP<>LOCS(3))*(LOCTMP<>LOCS(4))THEN 5200
5150 CALL CLEAR
5160 PRINT "You fell into a PIT!!!": : : : : :
5170 GOSUB 8000
5180 WIN=-1
5190 RETURN
5200 IF (LOCTMP<>LOCS(5))*(LOCTMP<>LOCS(6))THEN 5260
5210 CALL CLEAR
5220 PRINT "BAT Snatch!!!!": : : : : :
5230 GOSUB 9000
5240 LOCTMP=RNDX(MAXROOMS)
5250 GOTO 5100
5260 RETURN
6000 REM INIT COLORS
6010 FOR I=1 TO 12
6020 CALL COLOR(I,16,1)
6030 NEXT I
6040 CALL SCREEN(13)
6050 RETURN
7000 REM ARROW SOUND
7010 CALL SOUND(60,-5,0)
7020 FOR S=3000 TO 5500 STEP 275
7030 A=.005*(S-3000)
```

Listing 2-7—cont. Wumpus Program

```
7040 CALL SOUND(-120,110,30,110,30,S,30,-8,A)
7050 NEXT S
7060 CALL SOUND(150,-6,0)
7070 RETURN
8000 REM PIT SOUND
8010 CALL SCREEN(2)
8020 FOR S=4500 TO 1150 STEP -50
8030 A=.00597*(4500-S)
8040 CALL SOUND(-200,S,A,S+75,A)
8050 NEXT S
8060 CALL SOUND(33,-5,4)
8070 CALL SOUND(370,-6,0)
8080 CALL SOUND(333,-6,4)
8090 CALL SOUND(303,-6,8)
8100 CALL SOUND(100,-6,12)
8110 CALL SOUND(67,-6,16)
8120 CALL SOUND(33,-7,18)
8130 CALL SOUND(33,-6,18)
8140 RETURN
9000 REM BAT SOUND
9010 CALL SCREEN(7)
9020 FOR S=4000 TO 3500 STEP -8
9030 A=.02*(4000-S)
9040 CALL SOUND(-220,S,A,S+100,A,136,30,-8,0)
9050 NEXT S
9060 CALL SOUND(90,-7,0)
9070 CALL SOUND(40,-6,0)
9080 CALL SCREEN(13)
9090 RETURN
10000 REM WINNER
10010 RESTORE 10020
10020 DATA 262,294,330,349,277,311,349,370,294,330,370
10030 DATA 392,330,370,415,440,349,392,440,466
10040 FOR I=1 TO 20
10050 READ NOTE(I)
10060 NEXT I
10070 FOR I=1 TO 20
10080 CALL SOUND(250,NOTE(I),0)
10090 NEXT I
10100 RETURN
11000 REM LOSER
11010 DATA 196,200,196,200
11020 DATA 196,200,147,1000
11030 RESTORE 11010
11040 FOR I=1 TO 4
11050 READ NOTE(I),TIME(I)
11060 NEXT I
11070 FOR J=1 TO 2
11080 FOR I=1 TO 4
11090 IF I>3 THEN 11120
11100 CALL SOUND(250,NOTE(I),0)
11110 GOTO 11160
11120 CALL SOUND(TIME(I)/4,NOTE(I),0)
11130 CALL SOUND(TIME(I)/4,NOTE(I),5)
11140 CALL SOUND(TIME(I)/4,NOTE(I),10)
11150 CALL SOUND(TIME(I)/4,NOTE(I),15)
11160 NEXT I
```

Listing 2-7—cont. Wumpus Program

```
11170 NEXT J
11180 RETURN
12000 REM DELAY FOR KEY STROKE
12010 PRINT :TAB(4);"HIT ANY KEY TO GO ON":
12020 CALL KEY(0,KY,S)
12030 IF S=0 THEN 12020
12040 RETURN
13000 REM CAVE LAYOUT DATA
13010 DATA 2,5,8,1,3,10,2,4,12,3,5,14,1,4,6
13020 DATA 5,7,15,6,8,17,1,7,9,8,10,18,2,9,11
13030 DATA 10,12,19,3,11,13,12,14,20,4,13,15,6,14,16
13040 DATA 15,17,20,7,16,18,9,17,19,11,18,20,13,16,19
```

3

Finances

This chapter gives you code for these finance-related programs:

- CHECKBOOK: Helps you balance your checkbook.
- FUTURE VALUE: Tells you the future value of your investment.
- LOAN: Helps you evaluate terms for several loans.
- MORTGAGE: Calculates an amortization table.
- PRESENT VALUE: How much you have to invest today to meet a future goal.

CHECKBOOK PROGRAM

What it does:

The Checkbook program helps you balance your checkbook. First, enter your current balance. Next, enter all the deposits you made since the last time you balanced your checkbook. Finally, enter check numbers and check amounts for all the checks you wrote since you last balanced your checkbook.

The program calculates your new balance as:

BALANCE = BEGINNING BALANCE + DEPOSITS − CHECKS

Once you've entered all your data, you have a chance to change it. You select the data to change from one of the program's menus.

You can also add more data to the data already entered.

What it shows you:

The Checkbook program is a rather simple menu-driven program that shows you how to use menus to get information.

Once your data is entered, you can change it by selecting from a menu. The program shows how you can decide whether or not the change should be made (it writes the current value and asks whether or not to change it).

What you can do:

You could make this program write your data to a tape/disk/ etc., and then read the stored data.

You could also write your check information to a printer. Currently, the program only writes to the screen. You might want to format a check register, if you own a printer.

Table 3-1. Checkbook Routines

Lines	Function
100-230	Initialize variables and screen.
240-340	Main processing.
1000-1040	Get beginning balance.
2000-2070	Get deposit information.
3000-3090	Get check number/check amount information.
4000-4330	Calculate balance and print totals.
5000-5220	Control changes to current information.
6000-6470	Add/delete/change check data.
7000-7090	Search for a check.
8000-8040	Delay routine.

Table 3-2. Checkbook Variables

Variable	Description
BEGBAL	Beginning balance.
CHKAMT(100)	Check amounts for the checks you enter.
CHKNUM$(100)	Check numbers for the checks you enter.
CHKS	Number of checks you enter.
CHKTOT	Total of all checks entered.
DEPOSITS	Total of all deposits entered.
ENDBAL	Calculated as BEGBAL+DEPOSITS−CHKTOT
PRT	Number of lines printed at the screen.

Listing 3-1. Checkbook Program

```
100 REM CHECKBOOK BALANCING PROGRAM
110 REM BALANCE = BEGINNING BALANCE + DEPOSITS - CHECKS
120 REM
130 DIM CHKAMT(100),CHKNUM$(100)
140 CALL CLEAR
150 PRINT "This is a checkbook":" balancing program."
160 PRINT : :"First, enter your current":
    " checkbook balance."
170 PRINT :"Next, enter your deposits."
180 PRINT :"Finally, enter your checks.":
    " Enter a check number and"
190 PRINT " a check amount."
200 PRINT
210 REM BEGIN PROCESSING
220 GOSUB 8000
230 REM GET BEGINNING BALANCE
240 GOSUB 1000
250 REM GET DEPOSITS
260 GOSUB 2000
270 REM GET CHECKS
280 GOSUB 3000
290 REM CALC AND PRINT TOTALS
300 GOSUB 4000
310 REM SEE IF ANY CHANGES
320 GOSUB 5000
330 CALL CLEAR
340 STOP
1000 REM GET THE BEGINNING BALANCE
1010 CALL CLEAR
1020 PRINT :
1030 INPUT "Enter your beginning balance -> ":BEGBAL
1040 RETURN
2000 REM GET THE DEPOSITS
2010 DEPOSITS=0
2020 PRINT :" Enter deposits (0 when done)."
2030 INPUT " Deposit -> ":DEP
2040 IF DEP=0 THEN 2070
2050 DEPOSITS=DEPOSITS+DEP
2060 GOTO 2030
2070 RETURN
3000 REM GET CHECKS
3010 PRINT :"Enter your check numbers and checks."
3020 PRINT "1234,95.90": :"Enter 0,0 when done."
3030 INPUT "CHK#,AMT -> ":C$,AMT
3040 IF (C$="0")*(AMT=0)THEN 3100
3050 CHKS=CHKS+1
3060 CHKNUM$(CHKS)=C$
3070 CHKAMT(CHKS)=AMT
3080 CHKTOT=CHKTOT+AMT
3090 GOTO 3030
3100 RETURN
4000 REM CALC BAL AND PRINT TOTALS
4010 ENDBAL=BEGBAL+DEPOSITS-CHKTOT
4020 CALL CLEAR
4030 PRINT :"Beginning balance was ";BEGBAL
4040 PRINT :" Deposits totalled ";DEPOSITS
4050 PRINT :" Checks totalled ";CHKTOT
4060 PRINT :"Final balance is ";ENDBAL
```

Listing 3-1—cont. Checkbook Program

```
4070 PRINT
4080 INPUT "Do you want to see details on the
     checks? (Y/N) ":Y$
4090 IF Y$="N" THEN 4120
4100 IF Y$<>"Y" THEN 4080
4110 GOSUB 4130
4120 RETURN
4130 REM PRINT CHECK DETAILS
4140 REM FIRST, PRINT HDGS
4150 GOSUB 4270
4160 REM NOW, PRINT CHKS
4170 FOR I=1 TO CHKS
4180 PRINT CHKNUM$(I),CHKAMT(I)
4190 PRT=PRT+1
4200 IF PRT<19 THEN 4230
4210 GOSUB 8000
4220 GOSUB 4270
4230 NEXT I
4240 IF PRT=0 THEN 4260
4250 GOSUB 8000
4260 RETURN
4270 REM PRINT HDGS
4280 PRT=0
4290 IF I=CHKS THEN 4330
4300 CALL CLEAR
4310 PRINT "CHECK NUM","CHECK AMT"
4320 PRINT
4330 RETURN
5000 REM MAKE CHANGES
5010 CALL CLEAR
5020 INPUT "Any changes? (Y/N) ":Y$
5030 IF Y$="N" THEN 5220
5040 IF Y$<>"Y" THEN 5020
5050 REM SELECT WHICH CHANGE
5060 PRINT :"Select item to change:": :
     " 1 Beginning Balance"
5070 PRINT " 2 Deposits":" 3 Checks":" 0 Done":
5080 PRINT
5090 INPUT "Enter selection -> ":CHG
5100 IF (CHG<0)+(CHG>3)THEN 5060
5110 ON CHG+1 GOSUB 5160,1000,2000,6000
5120 INPUT "More changes (Y/N) ":Y$
5130 IF Y$="N" THEN 5160
5140 IF Y$<>"N" THEN 5120
5150 GOTO 5060
5160 REM DONE WITH CHANGES
5170 GOSUB 4000
5180 INPUT "More changes (Y/N) ":Y$
5190 IF Y$="N" THEN 5220
5200 IF Y$<>"Y" THEN 5180
5210 GOTO 5060
5220 RETURN
6000 REM CHANGE CHECKS
6010 PRINT :"Select type of change"
6020 PRINT :" 1 All new checks":" 2 Add more checks"
6030 PRINT " 3 Change an existing check":
     " 4 Erase an existing check"
6040 PRINT " 0 Done"
```

Listing 3-1—cont. Checkbook Program

```
6050 INPUT "Enter your selection -> ":SEL
6060 IF (SEL<0)*(SEL>4)THEN 6010
6070 ON SEL+1 GOSUB 6900,6130,6180,6210,6310
6080 INPUT "More check Changes (Y/N) ":Y$
6090 IF Y$="N" THEN 6120
6100 IF Y$<>"Y" THEN 6080
6110 GOTO 6010
6120 RETURN
6130 REM ALL NEW CHECKS
6140 CHKS=0
6150 CHKTOT=0
6160 GOSUB 3000
6170 RETURN
6180 REM ADD MORE CHECKS
6190 GOSUB 3000
6200 RETURN
6210 REM CHANGE A CHECK
6220 INPUT "Change which check number":C$
6230 IF C$="" THEN 6490
6240 GOSUB 7000
6250 IF FOUND=0 THEN 6310
6260 PRINT "Check amount is ";CHKAMT(FOUND)
6270 INPUT "Enter check number, check amount":C$,AMT
6280 CHKNUM$(FOUND)=C$
6290 CHKAMT(FOUND)=AMT
6300 GOTO 6220
6310 RETURN
6320 REM DELETE A CHECK
6330 INPUT "Erase which check number":C$
6340 IF C$="" THEN 6470
6350 GOSUB 7000
6360 IF FOUND=0 THEN 6470
6370 PRINT "Check amount is ";CHKAMT(FOUND)
6380 INPUT " Erase it (Y/N) ":Y$
6390 IF Y$="N" THEN 6330
6400 IF Y$<>"Y" THEN 6370
6410 FOR I=FOUND TO CHKS-1
6420 CHKNUM$(I)=CHKNUM$(I+1)
6430 CHKAMT(I)=CHKAMT(I+1)
6440 NEXT I
6450 CHKS=CHKS-1
6460 GOTO 6330
6470 RETURN
7000 REM SEARCH FOR A CHECK
7010 FOUND=0
7020 FOR I=1 TO CHKS
7030 IF CHKNUM$(I)<>C$ THEN 7060
7040 FOUND=I
7050 GOTO 7090
7060 NEXT I
7070 PRINT :"Check # ";C$;" not found."
7080 PRINT
7090 RETURN
8000 REM CAUSE DELAY
8010 PRINT :"Press any key to go on"
8020 CALL KEY(0,K,ST)
8030 IF ST=0 THEN 8020
8040 RETURN
```

FUTURE VALUE PROGRAM

What it does:

Future value is an expression of the value of an asset, growing and accumulating interest at a specified rate, over some future period.

There are three future value-oriented computations included in this program:

- the future value of an investment
- the future value of an annuity
- the annuity value of an investment.

The *future value of an investment* is a fancy term for a simple interest calculation. It tells you how much an investment made today, at a certain interest rate, will be worth after a specified number of years. To compute this, you must supply:

- the amount of the investment
- the average annual interest rate over the period of the investment
- the length of the investment in years
- the number of times per year that interest is compounded.

An annuity is a regular, yearly payment of a fixed sum of money. The *future value of an annuity* is the value, at some future date, of a regular, recurring, yearly investment. For example, suppose you want to invest $2,000 each year for 25 years. You expect to get 8% interest on your investment over the 25-year period. The question is: How much money will you have at the end of the 25 years?

In order to calculate the future value of an annuity, you must specify:

- the yearly annuity investment
- the expected annual interest rate
- the life of the annuity investment in years
- the number of times per year that an equal partial investment in the annuity is made.

Note that if you specify multiple investments per year, the payment period will also be considered an *interest compounding period.* That is, if you say you are making an annual investment of $2,000 in four equal payments (one per quarter), a consequence of the mathematics of the interest calculation results in a four-times-per-year interest compounding.

If your investment does compound interest on a daily, weekly, monthly, quarterly, or semi-annual basis, you can try to determine the effective annual interest rate (which includes the effects of com-

pounding). Except for very large amounts of money, the difference is not all that great, and given the uncertainty of interest projections over long periods of time, probably not worth worrying about.

Another question concerning annuities is how much you can withdraw from it if it must last for a certain period of time. This is the *annuity value of an investment.* For example, suppose you have $100,000 to invest which you would like to see pay you an annuity (yearly payment) for 10 years. Supposing your investment can yield an average of 10%, how much can you take as an annuity payment, and still have the investment last for 10 years?

To obtain the annuity payment, you must supply:

- the amount to be invested
- the desired life of the annuity in years
- the expected annual interest rate
- the number of times per year that an equal partial payment of the annuity is made.

The same considerations of partial payments as we discussed previously under the future value of an annuity computation apply to this computation.

We often think of annuities as a province of the rich. But an Individual Retirement Account can also be looked upon as an annuity. If you have (or are building) an IRA, you can use this program and the Present Value program elsewhere in this book, to get a good idea of how much will accumulate in that IRA and how long it will last at various rates of withdrawal (various annuity payments).

What it shows you:

This is a straightforward financial computation program whose structure and methods can be applied to many similar situations.

What you can do:

Try combining this program with the Present Value program. (HINT: If you have Extended BASIC, use the MERGE feature to make this easier.)

Table 3-3. Future Value Routines

Lines	Function
100-250	Main program control code.
1000-1190	Computes the future value of an investment.
2000-2200	Computes the future value of an annuity.
3000-3190	Computes the annuity value of an investment.

Table 3-4. Future Value Variables

Variables	Description
CMPD	The number of times per year interest is compounded in the lump sum calculation.
CMPDINT	The interest rate at each compounding period.
FV	The future value of an investment.
INTEREST	The stated yearly interest rate.
INVEST	The amount of the investment.
PART	The number of times per year a partial payment is made.
PARTINT	The partial interest rate for each payment period.
PARTPAY	The amount of a partial payment (or interest).
PAYMENT	The yearly annuity payment (or investment).
PERIOD	The number of partial periods in the term of the investment.
YEARS	The length of the investment in years.
USER FUNCTION	
RD(X)	Rounds the value X to two decimal places (n.nn).

Listing 3-2. Future Value Program

```
100 REM FUTURE VALUE PROGRAM
110 DEF RD(X)=INT(X*100+.5)/100
120 CALL CLEAR
130 PRINT : : : : : :TAB(2);"FUTURE VALUE CALCULATOR"
140 FOR I=1 TO 200
150 NEXT I
160 PRINT : : :"1. Future value of":"     a lump sum
    investment."
170 PRINT "2. Future value of":"     an annuity."
180 PRINT "3. Annuity value of ":"     an
    investment.":"4. END": : :
190 INPUT "Your choice->":ANS
200 IF (ANS>0)*(ANS<5)THEN 230
210 CALL SOUND(200,131,2,262,4,-2,3)
220 GOTO 160
230 ON ANS GOSUB 1000,2000,3000,250
240 GOTO 160
250 STOP
1000 REM FV OF A LUMP SUM
1010 CALL CLEAR
1020 PRINT "Enter the amount"
1030 INPUT "  to be invested->":INVEST
1040 PRINT :"Enter the yearly interest"
1050 INPUT "   rate(%)->":INTEREST
1060 PRINT :"Enter the period":" of the investment"
1070 INPUT "   in years->":YEARS
1080 PRINT :"How many times per year"
1090 INPUT "is interest compounded->":CMPD
1100 INTEREST=INTEREST/100
1110 CMPDINT=INTEREST/CMPD
1120 PERIOD=YEARS*CMPD
1130 FV=INVEST*((1+CMPDINT)^PERIOD)
1140 PRINT : : :"Investment of $";STR$(INVEST)
1150 PRINT "for ";STR$(YEARS);" years"
1160 PRINT "at ";STR$(INTEREST*100);"% interest"
```

Listing 3-2—cont. Future Value Program

```
1170 PRINT "compounded ";STR$(CMPD);" times per year"
1180 PRINT "yields $";STR$(RD(FV))
1190 RETURN
2000 REM FV OF AN ANNUITY
2010 CALL CLEAR
2020 PRINT "Enter the"
2030 INPUT "  yearly payment->":PAYMENT
2040 PRINT :"Enter the yearly interest"
2050 INPUT "   rate(%)->":INTEREST
2060 PRINT :"Enter the period":" of the annuity
     investment"
2070 INPUT "   in years->":YEARS
2080 PRINT :"How many times per year":" is a partial"
2090 INPUT "   payment made->":PART
2100 INTEREST=INTEREST/100
2110 PARTINT=INTEREST/PART
2120 PERIOD=YEARS*PART
2130 PARTPAY=PAYMENT/PART
2140 FV=(PARTPAY*((((1+PARTINT)
     ^(PERIOD+1))-1)/PARTINT))-PARTPAY
2150 PRINT : : :"Annual investment of $";STR$(PAYMENT)
2160 PRINT "for ";STR$(YEARS);" years"
2170 PRINT "at ";STR$(INTEREST*100);"% interest"
2180 PRINT "made in ";STR$(PART):"   equal sums per
     year"
2190 PRINT "yields $";STR$(RD(FV))
2200 RETURN
3000 REM PAYMENT OVER TIME
3010 CALL CLEAR
3020 PRINT "Enter the amount of"
3030 INPUT "  the investment->":INVEST
3040 PRINT :"Enter the yearly interest"
3050 INPUT "   rate(%)->":INTEREST
3060 PRINT :"Enter the desired period"
3070 INPUT "   in years->":YEARS
3080 PRINT :"How many times per year":" is a partial"
3090 INPUT "   payment made->":PART
3100 INTEREST=INTEREST/100
3110 PARTINT=INTEREST/PART
3120 PERIOD=YEARS*PART
3130 PAYMENT=INVEST/(1-(1/(1+PARTINT)^PERIOD))*PARTINT
3140 PRINT : : :"An investment of $";STR$(INVEST)
3150 PRINT "for ";STR$(YEARS);" years"
3160 PRINT "at ";STR$(INTEREST*100);"% interest"
3170 PRINT "with payments made in ";STR$(PART):
     "   equal sums per year"
3180 PRINT "yields an annual":"  payment of $";
     STR$(RD(PAYMENT*PART))
3190 RETURN
```

LOAN PROGRAM

What it does:

The Loan program calculates payments and total interest paid for up to 15 different loan combinations. You enter this loan data:

- Principal amount (up to $999,999)
- Loan term (up to 30 years)
- Interest rate (up to 99.99%)

The program prints the monthly payment and total interest for each entry. Then you choose from this menu:

1. Enter loan data.
2. Change loan data.
3. Print loan report.
4. STOP.

What it shows you:

The Loan program is rather straightforward. It asks you for information and calculates the payment and total interest. You'll notice that it checks the answers to be sure that you haven't entered something out of range. For example, you can't calculate the payment for a negative interest rate or a negative principal amount.

You get the chance to change loan data during the processing. Notice that the program tells you what the current data is before you change the information.

The information printed in the loan report uses the STR$ function for printing numeric data (principal, term, payment, total interest). This makes it easier to format the report. (TI BASIC puts a space around numeric data when it is printed.)

What you can do:

One thing that you may want to do is increase the number of loans processed to more than the current 15. Change the array dimensions at statement 120 and increase the value of MAXLOANS.

You can change the loan payment calculation to include an extra monthly payment. This means that you would pay off the loan much faster (even with a small extra payment). Look at the Mortgage program to see how an extra payment is handled.

At this time, the program only prints to the screen. If you have a printer, it would be nice to make the program print to it. The easiest

way to do this is to add another menu choice at statements 240 through 280 and the appropriate GOSUB entry at statement 310. Then write a routine to print the data.

Since there's a current limit of 15 loans, there's no need to check if you're writing more information than fits on one screen. If you increase the number of loans to more than 20, you should insert code in section 3000 to check how many lines are written. The loan program shows you a very simple way to pause every 20 lines. Other programs show you more complex techniques which are most useful if you decide to print heading information on each new screen's data.

Table 3-5. Loan Routines

Lines	Function
100-230	Initialize variables and screen.
240-320	Main menu processing.
1000-1280	Get loan data (principal, term, interest rate).
2000-2560	Change the loan data.
3000-3120	Print the loan information (principal, term, monthly payment, total interest).
4000-4220	Calculate the loan payment and total interest.
5000-5040	Delay function.

Table 3-6. Loan Variables

Variable	Description
AMOUNT	Temporary variable used in calculations.
AMT(15)	Principal amount for each loan.
BAL	Loan balance—used in calculating total interest for the loan.
INRATE	Interest rate scaled by 100 (INTRST/100) so that it can be used in calculations.
INTPMT	Calculated interest payment for a monthly payment.
INTRST	Interest rate entered.
LOANINT(15)	Total interest for each loan.
LOANS	Number of loans being processed.
MAXLOANS	Maximum number of loans that can be processed (currently 15).
MTH	Monthly interest rate.
NUMPMTS	Number of payments for the loan.
PAYMENT	Calculated monthly payment.
PMT(15)	Monthly payment for each loan.
PRPMT	Calculated principal payment for a monthly payment.
RATE(15)	Interest rate for each loan.
TOTINT	Total interest paid over the term of the loan.
YEARS(15)	Term for each loan.

Listing 3-3. Loan Program

```
100 REM THIS PROGRAM HELPS YOU TO EVALUATE LOANS
110 REM
120 DIM AMT(15),PMT(15),LOANINT(15),YEARS(15),RATE(15)
130 MAXLOANS=15
140 CALL CLEAR
150 PRINT "** LOAN PAYMENT ANALYZER **": : : : :
160 PRINT "This program helps you to":
    "analyze loan options."
170 PRINT : :"You enter the loan term,":
    "   (up to 30 yrs)"
180 PRINT "principal,":"   (up to 999,999)"
190 PRINT "and interest rate":"   (up to 99.99%).": :
200 PRINT "The program tells you":
    "the monthly payments."
210 PRINT : :
220 GOSUB 5000
230 CALL CLEAR
240 PRINT : :"You can:":" 1 Enter loan data":
    " 2 Change loan data"
250 PRINT " 3 Print loan report":" 4 STOP"
260 PRINT
270 INPUT "Your choice -> ":ANS
280 IF (ANS<1)+(ANS>4)THEN 240
290 IF ANS<4 THEN 310
300 STOP
310 ON ANS GOSUB 1000,2000,3000
320 GOTO 240
1000 REM GET LOAN DATA
1010 LOANS=0
1020 LOANS=LOANS+1
1030 PRINT :"Enter loan amount"
1040 INPUT " (0 to end) -> ":AMOUNT
1050 IF AMOUNT<>0 THEN 1080
1060 LOANS=LOANS-1
1070 RETURN
1080 IF (AMOUNT>0)*(AMOUNT<=999999)THEN 1120
1090 PRINT "The loan amount must be":
     "more than 0 and less":"than 999,999"
1100 PRINT
1110 GOTO 1030
1120 AMT(LOANS)=AMOUNT
1130 INPUT "How many years -> ":YRS
1140 IF (YRS>0)*(YRS<=99)THEN 1180
1150 PRINT "The loan term must be":
     "more than 0 and less":"than 99 years."
1160 PRINT
1170 GOTO 1130
1180 YEARS(LOANS)=YRS
1190 INPUT "At what interest rate -> ":INTRST
1200 IF (INTRST>0)*(INTRST<=99.99)THEN 1240
1210 PRINT "The interest rate must be":
     "more than 0 and less":"than 99.99%"
1220 PRINT
1230 GOTO 1190
1240 RATE(LOANS)=INTRST
1250 GOSUB 4000
1260 PMT(LOANS)=PAYMENT
1270 LOANINT(LOANS)=TOTINT
```

Listing 3-3—cont. Loan Program

```
1280 GOTO 1020
2000 REM CHANGE LOAN DATA
2010 IF LOANS>0 THEN 2040
2020 PRINT :"You don't have any data to":"change yet."
2030 RETURN
2040 PRINT "Change which loan"
2050 INPUT "(0 to end) -> ":CHG
2060 IF CHG=0 THEN 2560
2070 IF (CHG>0)*(CHG<=LOANS)THEN 2100
2080 PRINT "You don't have a";CHG;" loan."
2090 GOTO 2040
2100 PRINT "You can change:":" 1 loan amount":
     " 2 loan term"
2110 PRINT " 3 interest rate":" 4 none": :
2120 INPUT "Your choice -> ":ANS
2130 IF ANS=4 THEN 2040
2140 IF (ANS>0)*(ANS<4)THEN 2160
2150 GOTO 2100
2160 ON ANS GOTO 2170,2320,2440
2170 REM CHANGE THE LOAN AMOUNT
2180 PRINT "The loan amount is:";AMT(CHG)
2190 INPUT "Change it (Y/N) -> ":Y$
2200 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 2220
2210 GOTO 2040
2220 INPUT "Enter new amount -> ":AMOUNT
2230 IF (AMOUNT>0)*(AMOUNT<=999999)THEN 2250
2240 GOTO 2220
2250 AMT(CHG)=AMOUNT
2260 INTRST=RATE(CHG)
2270 YRS=YEARS(CHG)
2280 GOSUB 4000
2290 LOANINT(CHG)=TOTINT
2300 PMT(CHG)=PAYMENT
2310 GOTO 2040
2320 REM CHANGE THE LOAN TERM
2330 PRINT "The loan term is:";YEARS(CHG)
2340 INPUT "Enter new term -> ":YRS
2350 IF (YRS>0)*(YRS<=99)THEN 2370
2360 GOTO 2340
2370 YEARS(CHG)=YRS
2380 INTRST=RATE(CHG)
2390 AMOUNT=AMT(CHG)
2400 GOSUB 4000
2410 LOANINT(CHG)=TOTINT
2420 PMT(CHG)=PAYMENT
2430 GOTO 2040
2440 REM CHANGE THE INTEREST RATE
2450 PRINT "The interest rate is:";RATE(CHG)
2460 INPUT "Enter new interest rate -> ":INTRST
2470 IF (INTRST>0)*(INTRST<=99.99)THEN 2490
2480 GOTO 2460
2490 YRS=YEARS(CHG)
2500 RATE(CHG)=INTRST
2510 AMOUNT=AMT(CHG)
2520 GOSUB 4000
2530 LOANINT(CHG)=TOTINT
2540 PMT(CHG)=PAYMENT
2550 GOTO 2040
```

Listing 3-3—cont. Loan Program

```
2560 RETURN
3000 REM PRINT LOAN REPORT
3010 IF LOANS>0 THEN 3040
3020 PRINT :"You don't have any data yet.": :
3030 RETURN
3040 CALL CLEAR
3050 PRINT "   LOAN   YR   PMT    INT":
     "== ====== == ====== ========"
3060 FOR I=1 TO LOANS
3070 PRINT STR$(I);TAB(4);STR$(AMT(I));TAB(11);
     STR$(YEARS(I));
3080 PRINT TAB(14);STR$(PMT(I));TAB(21);
     STR$(LOANINT(I))
3090 PRINT
3100 NEXT I
3110 GOSUB 5000
3120 RETURN
4000 REM CALCULATE THE PAYMENT AND TOTAL INTEREST
4010 REM CALCULATE MONTHLY PAYMENT
4020 INRATE=INTRST/100
4030 NUMPMTS=12*YRS
4040 MTH=INRATE/12
4050 PAYMENT=(MTH*AMOUNT)/(1-1/((MTH+1)^(NUMPMTS)))
4060 PAYMENT=INT(PAYMENT*100)/100
4070 REM NOW CALCULATE THE TOTAL INTEREST PAYMENTS
4080 BAL=AMOUNT
4090 TOTINT=0
4100 FOR I=1 TO NUMPMTS
4110 INTPMT=INT(BAL*MTH*100)/100
4120 TOTINT=TOTINT+INTPMT
4130 REM IF REMAINING BAL <0 THEN THIS IS THE LAST
     PAYMENT
4140 IF BAL+INTPMT<=PAYMENT THEN 4190
4150 PRPMT=PAYMENT-INTPMT
4160 BAL=BAL-PRPMT
4170 NEXT I
4190 PRINT :"For a ";AMOUNT;" loan":"at";INTRST;
     "% interest"
4200 PRINT "over";YRS;" years":"you'll pay";PAYMENT;
     " each month ";
4210 PRINT "and a total of";TOTINT;" in interest.": :
4220 RETURN
5000 REM PAUSE
5010 PRINT "Press any key to continue.";
5020 CALL KEY(0,K,S)
5030 IF S=0 THEN 5020
5040 RETURN
```

MORTGAGE PROGRAM

What it does:

The Mortgage program calculates an amortization table for a fixed rate mortgage. You can print the entire table or just the year total information either to a printer or to the screen.

This program calculates an amortization table for a mortgage that has:

- Principal from $.01 to $999,999
- Interest rate from 1% to 99.99%
- Term of 1 month to 40 years
- Beginning mortgage month of 1 through 12
- Beginning mortgage year of 0 through 99
- A fixed additional monthly payment applied to reducing the principal.

After you enter the first five of these variables, the program calculates the monthly mortgage payment (principal + interest). You then have a chance to add an additional payment to this calculated payment. The additional payment is applied to the principal and lets you pay off your mortgage early.

You now choose from this menu:

1. Display totals only
2. Display entire table
3. Print totals only
4. Print entire table
5. Display and print totals only
6. Display and print entire table
7. Change data
8. STOP.

When you choose display, you'll see your information at your television screen. If you have a printer, you can also print the data.

Any of the mortgage variables (principal amount, term, etc.) can be changed and the amortization table recalculated. We tried redoing the table with varying additional payments to see when a high interest mortgage (like the ever-popular 16% and up) would be paid off. The amount of interest payments that can be saved is staggering.

What it shows you:

The Mortgage program shows you a tricky way to write to two different devices (screen and printer) using a FOR loop. We assigned a value to the HIFILE and LOFILE variables, depending on what device was used.

The FOR loops used for printing go from LOFILE to HIFILE. A file number of zero in a PRINT statement (PRINT #0) writes to the screen. A file number of one (PRINT #1) writes to the printer. (The printer file is opened in line 710.)

Using this technique, it's not necessary to check where to write. The FOR loops do it all automatically. It is necessary to CLOSE the print file before trying to reOPEN it for later printing.

Since the amortization table, and even just the totals for a long mortgage, will quickly fill up a screen, we put in code to stop writing to the screen every 20 or so lines. This uses the ever-popular Delay routine that you'll see in most of the programs in this book.

What you can do:

One thing you may have to do is fix the OPEN statement at line 710 to reflect the characteristics of your own printer. Just look at the instruction book that came with your printer to see what baud rate (BA = 9600 in our example) you should use. You may also have to set up some other characteristics, depending on which printer you have.

If you're really ambitious, you can modify the program to calculate a variable-rate mortgage. We felt that you should have some challenges.

You may also want to reformat the way the amortization table and totals appear. We tried to fit it onto the screen as neatly as possible. Rearrange it to meet your own needs.

Table 3-7. Mortgage Routines

Lines	Function
100-220	Initialize variables and screen.
230-840	Main processing.
1000-1170	Calculate monthly payment.
2000-2270	Change mortgage data.
3000-3150	Calculate amortization table.
4000-4290	Print routine driver.
5000-5310	Print year total information.
6000-6480	Print entire table.
7000-7040	Delay function.

Table 3-8. Mortgage Variables

Variable	Description
ACCINT	Accumulated interest for the mortgage.
BAL	Remaining balance. Used in calculating the table.
BEGMTH	Beginning month for the mortgage.
BEGYR	Beginning year for the mortgage.
CALCPMT	Calculated monthly payment.
FINPMT	Final payment. Adjusted for any rounding errors.
HIFILE	Used in printing to the printer/screen.
INTEREST	Mortgage interest rate (up to 99.99%).
INTPMT(480)	Interest payment for each monthly payment.
INTYR	Total interest paid in one year.
LOFILE	Used in printing to the printer/screen.
LSTPMT	Number of the last actual payment (e.g., 360 for 30 years, 349 for 30-year mortgage with extra monthly payment made).
MTH	Monthly interest rate (RATE/12).
NUMPMTS	Number of payments in the term (12*YEARS).
PL	Number of lines printed at the printer.
PRIN	Amount mortgaged (up to $999,999).
PRINYR	Total principal paid in one year.
PRPMT(480)	Principal payment for each monthly payment.
RATE	Scaled mortgage interest rate (INTEREST/100).
RD	Rounding function that rounds to two decimal places (nn.nn).
TL	Number of lines printed at the screen.
TOTFLG	Indicator used to print totals only (TOTFLG=1) or entire amortization table (TOTFLG=0).
YEARS	Mortgage term (up to 40 years).

Listing 3-4. Mortgage Program

```
100 REM CALCULATE FIXED RATE MORTGAGE PMTS
110 OPTION BASE 1
120 DIM INTPMT(480),PRPMT(480)
130 DEF RD(X)=INT(X*100+.5)/100
140 CALL CLEAR
150 PRINT TAB(10);"MORTGAGE";TAB(8);"AMORTIZATION";
    TAB(10);"PROGRAM"
160 PRINT : : : : : :
170 PRINT TAB(6);"YOU CAN CALCULATE":TAB(5);
    "A MORTGAGE TABLE FOR:"
180 PRINT :" PRINCIPAL UP TO $999,999":
    " INTEREST RATE UP TO 99.99%"
190 PRINT " FOR A MAXIMUM OF 40 YEARS."
200 GOSUB 7000
210 LSTPMT=0
220 CALL CLEAR
230 INPUT "ENTER BEGINNING MONTH(NN)->":BEGMTH
240 IF (BEGMTH<1)+(BEGMTH>12)THEN 230
250 INPUT "ENTER BEGINNING YEAR(NN)->":BEGYR
260 IF (BEGYR<0)+(BEGYR>99)THEN 250
270 INPUT "ENTER PRINCIPAL AMOUNT->":PRIN
280 IF (PRIN<0)+(PRIN>999999)THEN 270
290 INPUT "ENTER INTEREST RATE(NN.NN)->":INTEREST
```

Listing 3-4—cont. Mortgage Program

```
300 IF (INTEREST<0.01)+(INTEREST>99.99)THEN 290
310 INPUT "ENTER TERM IN YEARS->":YEARS
320 IF (YEARS<0.01)+(YEARS>40)THEN 310
330 PRINT
340 GOSUB 1000
350 INPUT "ANY CHANGES (Y/N)-> ":Y$
360 PRINT
370 IF Y$<>"Y" THEN 400
380 GOSUB 2000
390 GOSUB 1000
400 INPUT "CALCULATE THE TABLE (Y/N)->":Y$
410 PRINT
420 IF Y$<>"Y" THEN 450
430 PRINT :"CALCULATING MORTGAGE TABLE"
440 GOSUB 3000
450 CALL CLEAR
460 PRINT "MORTGAGE PROGRAM OPTIONS:": :
    " 1 DISPLAY TOTALS ONLY":
470 PRINT " 2 DISPLAY ENTIRE TABLE":
    " 3 PRINT TOTALS ONLY"
480 PRINT " 4 PRINT ENTIRE TABLE":
    " 5 DISPLAY & PRINT TOTALS":"   ONLY"
490 PRINT " 6 DISPLAY & PRINT ENTIRE":"   TABLE":
    " 7 CHANGE DATA":" 8 STOP": :
500 INPUT "YOUR CHOICE -> ":ANS
510 IF (ANS<1)+(ANS>8)THEN 450
520 IF ANS<>8 THEN 540
530 STOP
540 IF LSTPMT>0 THEN 570
550 PRINT :"NO ENTRIES IN TABLE."
560 GOTO 350
570 ACCINT=0
580 BAL=PRIN
590 INTYR=0
600 PRINYR=0
610 PRTYR=1900+BEGYR-1
620 MONTH=BEGMTH-1
630 YR=0
640 LOFILE=0
650 HIFILE=0
660 FIRST=1
670 IF (ANS<3)+(ANS>4)THEN 690
680 LOFILE=1
690 IF (ANS<3)THEN 720
700 HIFILE=1
710 OPEN #1:"RS232.BA=9600",OUTPUT
720 ON ANS GOTO 790,820,790,820,790,820,730
730 IF HIFILE=0 THEN 380
740 CLOSE #1
750 GOTO 380
760 IF HIFILE=0 THEN 450
770 CLOSE #1
780 GOTO 450
790 REM TOTALS ONLY
800 GOSUB 4000
810 GOTO 760
820 REM ENTIRE TABLE
830 GOSUB 6000
```

Listing 3-4—cont. Mortgage Program

```
840 GOTO 760
1000 REM CALCULATE MONTHLY PAYMENT
1010 RATE=INTEREST/100
1020 NUMPMTS=12*YEARS
1030 MTH=RATE/12
1040 CALCPMT=(MTH*PRIN)/(1-1/((MTH+1)^(NUMPMTS)))
1050 CALCPMT=RD(CALCPMT)
1060 PRINT :"MONTHLY PAYMENT IS :";CALCPMT
1070 PRINT
1080 PMT=CALCPMT
1090 PRINT "DO YOU WANT TO MAKE AN":
     " EXTRA PAYMENT EACH"
1100 INPUT " MONTH?(Y/N)->":Y$
1110 PRINT
1120 IF Y$<>"Y" THEN 1160
1130 PRINT :"ENTER EXTRA MONTHLY"
1140 INPUT "  PAYMENT ->":ADDPMT
1150 PMT=CALCPMT+ADDPMT
1160 PRTPMT=PMT
1170 RETURN
2000 REM CHANGE SOME DATA
2010 CALL CLEAR
2020 PRINT "DATA TO CHANGE:": :" 1 BEGINNING MONTH":
     " 2 BEGINNING YEAR"
2030 PRINT " 3 PRINCIPAL AMOUNT":" 4 INTEREST RATE":
     " 5 TERM":" 6 DONE": :
2040 INPUT "CHANGE WHICH ITEM -> ":ANS
2050 IF (ANS<1)+(ANS>6)THEN 2000
2060 ON ANS GOTO 2070,2110,2150,2190,2230,2270
2070 PRINT "BEGINNING MONTH: ";BEGMTH:
     "ENTER NEW BEGINNING"
2080 INPUT "  MONTH (NN) -> ":BEGMTH
2090 IF (BEGMTH<1)+(BEGMTH>12)THEN 2070
2100 GOTO 2040
2110 PRINT "BEGINNING YEAR: ":BEGYR:
     "ENTER NEW BEGINNING"
2120 INPUT "  YEAR (NN) -> ":BEGYR
2130 IF (BEGYR<0)+(BEGYR>99)THEN 2110
2140 GOTO 2040
2150 PRINT "PRINCIPAL AMOUNT: ";PRIN:
     "ENTER NEW PRINCIPAL"
2160 INPUT " AMOUNT -> ":PRIN
2170 IF (PRIN<0)+(PRIN>999999)THEN 2150
2180 GOTO 2040
2190 PRINT "INTEREST RATE: ";INTEREST:
     "ENTER NEW INTEREST"
2200 INPUT "  RATE (NN.NN) -> ":INTEREST
2210 IF (INTEREST<.01)+(INTEREST>99.99)THEN 2190
2220 GOTO 2040
2230 PRINT "MORTGAGE TERM: ";YEARS
2240 INPUT "ENTER NEW TERM -> ":YEARS
2250 IF (YEARS<0)+(YEARS>40)THEN 2230
2260 GOTO 2040
2270 RETURN
3000 REM CALCULATE TABLE
3010 BAL=PRIN
3020 FOR I=1 TO NUMPMTS
3030 LSTPMT=I
```

Listing 3-4—cont. Mortgage Program

```
3040 IF INT(I/12)<>I/12 THEN 3060
3050 PRINT "CALCULATING YEAR ";INT(I/12)
3060 INTPMT(I)=RD(BAL*MTH)
3070 REM IF REMAINING BAL <0 THEN THIS IS THE LAST
     PAYMENT
3080 IF BAL+INTPMT(I)<=PMT THEN 3130
3090 PRPMT(I)=PMT-INTPMT(I)
3100 BAL=BAL-PRPMT(I)
3110 NEXT I
3120 BAL=BAL+PRPMT(LSTPMT)
3130 FINPMT=BAL+INTPMT(LSTPMT)
3140 PMT=BAL
3150 RETURN
4000 REM PRINT/DISPLAY TOTALS ONLY
4010 TOTFLG=1
4020 GOSUB 4210
4030 FOR I=1 TO LSTPMT-1
4040 MONTH=MONTH+1
4050 INTYR=INTYR+INTPMT(I)
4060 ACCINT=ACCINT+INTPMT(I)
4070 PRINYR=PRINYR+PRPMT(I)
4080 BAL=BAL-PRPMT(I)
4090 IF MONTH<>12 THEN 4140
4100 YR=YR+1
4110 GOSUB 5000
4120 INTYR=0
4130 PRINYR=0
4140 NEXT I
4150 FINPMT=BAL+INTPMT(LSTPMT)
4160 GOSUB 5190
4170 GOSUB 5250
4180 IF LOFILE=1 THEN 4200
4190 GOSUB 7000
4200 RETURN
4210 FOR J=LOFILE TO HIFILE
4220 CALL CLEAR
4230 PRINT #J:CHR$(140);TAB(6);"MORTGAGE TOTALS":
     "$";PRIN;
4240 PRINT #J:" AT  ";INTEREST;"%":"FOR ";YEARS;
     " YRS: $";RD(PRTPMT);"/MO"
4250 PRINT #J:"BEGINNING";BEGMTH;"/";INT(1900+BEGYR):
4260 NEXT J
4270 TL=4
4280 PL=4
4290 RETURN
5000 REM PRINT YEAR TOTAL LINE
5010 PRTYR=PRTYR+1
5020 MONTH=0
5030 FOR J=LOFILE TO HIFILE
5040 PRINT #J: :"YEAR ";YR;"  ";PRTYR:"  INTEREST: ";
5050 PRINT #J:TAB(14);RD(INTYR):"  PRINCIPAL: ";
     TAB(14);RD(PRINYR)
5060 FIRST=0
5070 PRINT #J:"  ACCUM INT:";TAB(14);RD(ACCINT):
     "  PRIN BAL: ";RD(BAL)
5080 NEXT J
5090 TL=TL+6
5100 IF (TL<20)+(LOFILE=1)THEN 5130
```

Listing 3-4—cont. Mortgage Program

```
5110 TL=0
5120 GOSUB 7000
5130 IF HIFILE=0 THEN 5240
5140 PL=PL+6
5150 IF PL<50 THEN 5240
5160 PRINT #1:CHR$(140);
5170 PL=0
5180 RETURN
5190 MONTH=MONTH+1
5200 FOR J=LOFILE TO HIFILE
5210 PRINT #J: :"FINAL PAYMENT: ";FINPMT
5220 PRINT #J:" MONTH:";MONTH:" INT:";
     RD(INTPMT(LSTPMT));" PRIN:";RD(BAL)
5230 NEXT J
5240 RETURN
5250 INTYR=INTYR+INTPMT(LSTPMT)
5260 ACCINT=ACCINT+INTPMT(LSTPMT)
5270 PRINYR=PRINYR+BAL
5280 BAL=0
5290 YR=YR+1
5300 GOSUB 5000
5310 RETURN
6000 REM PRINT ENTIRE TABLE
6010 TOTFLG=0
6020 FIRST=1
6030 GOSUB 6400
6040 FOR I=1 TO LSTPMT-1
6050 MONTH=MONTH+1
6060 INTYR=INTYR+INTPMT(I)
6070 ACCINT=ACCINT+INTPMT(I)
6080 PRINYR=PRINYR+PRPMT(I)
6090 BAL=BAL-PRPMT(I)
6100 GOSUB 6230
6110 IF MONTH<>12 THEN 6160
6120 YR=YR+1
6130 GOSUB 5000
6140 INTYR=0
6150 PRINYR=0
6160 NEXT I
6170 FINPMT=BAL+INTPMT(LSTPMT)
6180 GOSUB 5190
6190 GOSUB 5250
6200 IF LOFILE=1 THEN 6220
6210 GOSUB 7000
6220 RETURN
6230 FOR J=LOFILE TO HIFILE
6240 IF (MONTH=1)+(FIRST=1)THEN 6250 ELSE 6280
6250 PRINT #J: :"MT    INT     PRIN     BALANCE":
6260 TL=TL+3
6270 PL=PL+3
6280 PRINT #J:STR$(MONTH);TAB(4);STR$(RD(INTPMT(I)));
6290 PRINT #J:TAB(12);STR$(RD(PRPMT(I)));TAB(20);
     STR$(RD(BAL))
6300 NEXT J
6310 TL=TL+1
6320 PL=PL+1
6330 IF FIRST=0 THEN 6350
6340 FIRST=0
```

Listing 3-4—cont. Mortgage Program

```
6350 IF (PL<55)+(HIFILE=0)THEN 6390
6360 PRINT #1:CHR$(140);
6370 PL=0
6380 FIRST=1
6390 RETURN
6400 REM PRINT HEADINGS
6410 CALL CLEAR
6420 FOR J=LOFILE TO HIFILE
6430 PRINT #J:CHR$(140);TAB(6);"MORTGAGE TABLE":
     "$";PRIN;
6440 PRINT #J:" AT  ";INTEREST;"%":"FOR ";YEARS;
     " YRS: $";RD(PRTPMT);"/MO"
6450 NEXT J
6460 TL=3
6470 PL=3
6480 RETURN
7000 REM DELAY
7010 PRINT : :"PRESS ANY KEY TO CONTINUE.";
7020 CALL KEY(0,K,S)
7030 IF S=0 THEN 7020
7040 RETURN
```

PRESENT VALUE PROGRAM

What it does:

Present value is an important concept in the management of money. It tells you how much you must invest now to achieve a certain goal in the future.

There are three present value-related computations included in this program:

- the present value of a lump sum future payment
- the present value of an annuity
- the period of a given annuity fund.

The *present value of a lump sum* future payment is actually the amount of money you must invest today, at a stated interest rate, to achieve that lump sum payment within a specified period of time. To calculate this present value you must supply:

- the desired lump sum payment
- the average annual interest rate over the period of the investment
- the length of the investment in years
- the number of times per year that interest is compounded.

An annuity is a regular, yearly, payment of a fixed sum of money. The *present value of an annuity* is the amount of money that must be invested to produce a particular annuity payment, at a stated interest

rate, over a specified period of years. For example, suppose you want to take $2,000 from an investment each year for 25 years. You expect to get 8% interest on your investment over the 25-year period. The question is: How much must you invest now to meet these requirements?

In order to calculate the present value of an annuity, you must specify:

- the desired yearly annuity payment
- the expected annual interest rate
- the life of the annuity in years
- the number of times per year that an equal partial payment of the annuity is made.

Note that if you specify multiple payments per year, the payment period will also be considered an *interest compounding period.* That is, if you say you are taking an annual payment of $2,000 in four equal installments (one per quarter), a consequence of the mathematics of the interest calculation results in a four-times-per-year interest compounding.

If your investment does compound interest on a daily, weekly, monthly, quarterly, or semiannual basis, you can try to determine the effective annual interest rate (which includes the effects of compounding). Except for very large amounts of money, the difference is not all that great, and given the uncertainty of interest projections over long periods of time, it's probably not worth worrying about.

A frequently encountered, and not so easily solved, problem with annuities is estimating *how long an annuity can be paid* from a given investment. For example, suppose you have $100,000 to invest which you would like to see pay you an annuity (yearly payment) of $15,000. Assume your investment can yield an average of 10%. How long can this investment, at this rate of interest, continue to produce a $15,000 annuity payment?

To obtain the life of an annuity, you must supply:

- the amount to be invested
- the desired yearly annuity payment
- the expected annual interest rate
- the number of times per year that an equal partial payment of the annuity is made.

The same considerations of partial payments that we discussed previously under the annuity computation apply to this computation.

We often think of annuities as a province of the rich. But an Indi-

vidual Retirement Account can also be looked upon as an annuity. If you have, or are building, an IRA, you can use this program, and the Future Value program elsewhere in this book, to get a good idea of how much will accumulate in that IRA and how long it will last at various rates of withdrawal (various annuity payments).

What it shows you:

This is a straightforward financial computation program whose structure and methods can be applied to many similar situations.

One unique element of this program is the binary search method used to locate the period of an annuity. The formula for an annuity is expressed as a value raised to the power of the period. There are several ways to solve for the exponent, but the simplest is to search for it, beginning with a high (in this case 100 years) and a low (1 period) guess.

Examine the code (statement range 4000). You can see that our guess is refined on each run through the calculation until it is within 0.5% of the actual period. The calculation is typically executed between five and ten times, producing a barely perceptible pause before the answer is printed.

We exit from the search when the result is within 0.5% of the actual value so that we don't have to hit the exact value (to fourteen decimal places) in order to exit from the routine. For most purposes, this is close enough.

What you can do:

Insert code into the period search routine to report the number of iterations required to find a result. Now increase the accuracy requirements (make LFA 0.9995 and HFA 1.0005) of the search and see the effect this has on the number of iterations.

Table 3-9. Present Value Routines

Lines	Function
100-250	Main program control code.
1000-1190	Computes the present value of a future lump sum payment.
2000-2200	Computes the present value of an annuity.
3000-3230	Handles calculation of the period of an annuity.
4000-4210	Performs a search for the period of an annuity.

Table 3-10. Present Value Variables

Variable	Description
CMPD	The number of times per year interest is compounded in the lump sum calculation.
CMPDINT	The interest rate at each compounding period.
HFA	The high side "fuzz" factor used to exist from the period search routine.
INTEREST	The stated yearly interest rate.
INVEST	The amount of the investment.
LFA	The low side "fuzz" factor used to exit from the period search routine.
LSUM	The desired lump sum payment.
PART	The number of times per year a partial payment is made.
PARTINT	The partial interest rate for each payment period.
PARTPAY	The amount of a partial payment.
PAYMENT	The yearly annuity payment
PERIOD	The number of partial periods in the term of the investment.
PH	"Period high" in the binary search routine.
PL	"Period low" in the binary search routine.
PV	The present value of a lump sum payment.
YEARS	The length of the investment in years.
USER FUNCTION	
RD(X)	Rounds the value X to two decimal places (n.nn).

Listing 3-5. Present Value Program

```
100 REM PRESENT VALUE PROGRAM
110 DEF RD(X)=INT(X*100+.5)/100
120 CALL CLEAR
130 PRINT : : : : : :TAB(2);"PRESENT VALUE CALCULATOR"
140 FOR I=1 TO 200
150 NEXT I
160 PRINT : : :"1. Present value of":
    "       a lump sum payment."
170 PRINT "2. Present value of":"       an annuity."
180 PRINT "3. Period of an annuity.":"4. END"
190 INPUT "Your choice->":ANS
200 IF (ANS>0)*(ANS<5)THEN 230
210 CALL SOUND(200,131,2,262,4,-2,3)
220 GOTO 160
230 ON ANS GOSUB 1000,2000,3000,250
240 GOTO 160
250 STOP
1000 REM PV OF A LUMP SUM
1010 CALL CLEAR
1020 PRINT "Enter the desired"
1030 INPUT "  lump sum payment->":LSUM
1040 PRINT :"Enter the yearly interest"
1050 INPUT "   rate(%)->":INTEREST
1060 PRINT :"Enter the period"
1070 INPUT "   in years->":YEARS
1080 PRINT :"How many times per year"
1090 INPUT "is interest compounded->":CMPD
1100 INTEREST=INTEREST/100
```

Listing 3-5—cont. Present Value Program

```
1110 CMPDINT=INTEREST/CMPD
1120 PERIOD=YEARS*CMPD
1130 PV=LSUM*(1/((1+CMPDINT)^PERIOD))
1140 PRINT : : :"Payment of $";STR$(LSUM)
1150 PRINT "after ";STR$(YEARS);" years"
1160 PRINT "at ";STR$(INTEREST*100);"% interest"
1170 PRINT "compounded ";STR$(CMPD);" times per year"
1180 PRINT "requires an investment":"      of $";
     STR$(RD(PV))
1190 RETURN
2000 REM PV OF AN ANNUITY
2010 CALL CLEAR
2020 PRINT "Enter the desired"
2030 INPUT "  yearly payment->":PAYMENT
2040 PRINT :"Enter the yearly interest"
2050 INPUT "   rate(%)->":INTEREST
2060 PRINT :"Enter the period"
2070 INPUT "   in years->":YEARS
2080 PRINT :"How many times per year":"  is a partial"
2090 INPUT "    payment made->":PART
2100 INTEREST=INTEREST/100
2110 PARTINT=INTEREST/PART
2120 PERIOD=YEARS*PART
2130 PARTPAY=PAYMENT/PART
2140 PV=PARTPAY*(1-(1/(1+PARTINT)^PERIOD))/PARTINT
2150 PRINT : : :"Annual payment of $";STR$(PAYMENT)
2160 PRINT "for ";STR$(YEARS);" years"
2170 PRINT "at ";STR$(INTEREST*100);"% interest"
2180 PRINT "paid in ";STR$(PART):
     "   equal payments per year"
2190 PRINT "requires an investment":"      of $";
     STR$(RD(PV))
2200 RETURN
3000 REM PERIOD OF ANNUITY
3010 CALL CLEAR
3020 PRINT "Enter the amount"
3030 INPUT "  invested->":INVEST
3040 PRINT :"Enter the desired"
3050 INPUT "  yearly payment->":PAYMENT
3060 PRINT :"Enter the yearly interest"
3070 INPUT "   rate(%)->":INTEREST
3080 PRINT :"How many times per year":" is a partial"
3090 INPUT "    payment made->":PART
3100 INTEREST=INTEREST/100
3110 PARTINT=INTEREST/PART
3120 PARTPAY=PAYMENT/PART
3130 GOSUB 4000
3140 YEARS=INT(10*PERIOD/PART+.5)/10
3150 PRINT : : :"Annual payment of $";STR$(PAYMENT)
3160 PRINT "from an investment":"      of $";
     STR$(INVEST)
3170 PRINT "at ";STR$(INTEREST*100);"% interest"
3180 PRINT "paid in ";STR$(PART):
     "   equal payments per year"
3190 IF YEARS=100 THEN 3220
3200 PRINT "lasts for ";STR$(YEARS);" years."
3210 RETURN
3220 PRINT "lasts for more than":"     100 years."
```

Listing 3-5—cont. Present Value Program

```
3230 RETURN
4000 REM SEARCH FOR PERIOD
4010 IF (INVEST*PARTINT)<PARTPAY THEN 4040
4020 PERIOD=100*PART
4030 RETURN
4040 X=PARTPAY/(PARTPAY-INVEST*PARTINT)
4050 Y=1+PARTINT
4060 HFA=1.005
4070 LFA=.995
4080 PH=100*PART
4090 PL=1
4100 PERIOD=PL
4110 IF Y^PERIOD>=X THEN 4210
4120 PERIOD=(PH+PL)/2
4130 Z=Y^PERIOD
4140 IF ((Z*HFA)>=X)*((Z*LFA)<=X)THEN 4210
4150 IF Z>X THEN 4180
4160 PL=PERIOD
4170 GOTO 4190
4180 PH=PERIOD
4190 PERIOD=(PH+PL)/2
4200 GOTO 4130
4210 RETURN
```

4

Home Management

This chapter gives you the code for these home-management programs:

- AUTO MAINTENANCE: Keep track of your auto repair history, costs, and routine maintenance schedule.
- FLOOR COVERING: Analyze the amount needed and cost of a variety of floor coverings.
- RECIPE CONVERTER: Increase or decrease the amount of ingredients needed for a recipe.
- WALLPAPER/PAINT: Analyze the amount needed and cost of paint or wallpaper.
- YARN ESTIMATOR: Find out how much yarn you need to cover your needlepoint canvas.

AUTO MAINTENANCE PROGRAM

What it does:

The auto maintenance program is designed to keep track of routine maintenance and other repairs to your automobiles. This is a menu-driven program that provides you with a list of things the program can do from which you choose what you want to do.

This makes the program very easy to operate. After you enter it, experiment with it until you become familiar with its operation.

The routine auto maintenance schedule includes:

- oil change
- tune up

- brake replacement
- tire rotation
- air conditioning maintenance
- fan belt replacement

To make this work, you must supply the recommended maintenance intervals for these activities. Intervals are expressed in both miles and months, depending on the item. You'll need the following routine maintenance service interval information:

- oil change interval in miles and months
- tune up interval in miles and months
- brake replacement interval in miles
- tire rotation interval in miles
- air conditioner maintenance interval in months
- fan belt check interval in months.

When you make a repair, you enter:

- the date of the repair
- the mileage at the time
- the cost of the repair.

Once you've entered the initial data, you can get reports listing:

- maintenance history by type
- "Service Now" report listing maintenance due now.

In addition to the routine maintenance, you can track other repairs to the vehicle. These nonroutine repairs include:

- description of the repair
- date the repair was made
- mileage at the time of the repair
- cost of the repair.

The program maintains a maximum of 10 repairs in each category. This is all that will fit into 16K of console RAM. When you enter the eleventh repair, the oldest one is removed from the list and discarded to make room for the new one.

All the information can be saved to a cassette, Wafertape, or disk file for later recall. If you're saving to or restoring from a cassette tape, use log filename CS1.

What it shows you:

This program includes a save/restore facility for writing and recalling data on tape or disk. If you are unfamiliar with input/output to

files, look at the routines in the 1000 and 5000 statements number range.

The user function DT$ illustrates the power of user-defined functions on the TI-99/4A. Creative application of user functions can save you much time and typing.

What you can do:

This program just fits into the 16K console memory. If you have expanded memory, you can easily add features, such as delete entry, insert entry, print total costs, or search repairs by keyword.

You may want to keep track of something other than what we've chosen. You can easily go through and systematically replace our choice with your own.

We've limited service history to ten items in each category. If you have more memory, you can increase this.

Table 4-1. Auto Maintenance Routines

Lines	Function
100-280	Main program control loop.
1000-1180	Restore auto log file.
3000-3490	Create new auto log.
5000-5160	Write auto log to file.
7000-8190	Print Service Now and Service History reports
9000-10230	Routine maintenance data update.
11000-11120	Other Repairs report.
13000-13180	Other repairs update.
15000-15140	Check service due by date.
16000-16060	Check service due by mileage.
18000-18040	Delay routine.
19000-19180	Initialize system variables and get current date and mileage.

Table 4-2. Auto Maintenance Variables

Variable	Description
AC(10,3)	Air conditioning maintenance information.
AUTONAME$	The name of the automobile.
BELTS(10,3)	Fan belt replacement information.
BRKS(10,3)	Brake replacement information.
COST	Cost of the current repair.
CURMILES	Current mileage on the vehicle.
CURMO	Current month.
CURYR	Current year.
DATE	Date of the repair.
DESC$(10)	Description of a nonroutine repair.
INM	Index into the INTV array.
INTV(8)	Service intervals for each routine maintenance item in miles, months, or both.
LAB$(6)	Labels of the various categories.
LN	Label number (in the LAB$ array).
LOGF$	Name of the log file for input or output.
MILES	Mileage used in calculations.
MONTH	Month used in calculations.
OIL(10,3)	Oil change information.
REP(10,3)	Other repairs information.
TIRE(10,3)	Tire rotation information.
TUP(10,3)	Tune up information.
YR	The year used in calculations.
USER FUNCTION	
DT$(X)	Creates a character string representation of the date (in X). The date is stored in the numeric variable passed as X coded as YYMM (i.e., 8310 for October 1983). The function produces a character string of the form MM/YY. For example, date 8310 becomes "10/83."

Listing 4-1. Auto Maintenance Program

```
100 OPTION BASE 1
110 REM AUTO MAINTENANCE LOG
120 DIM OIL(10,3),TUP(10,3),TIRE(10,3),LAB$(6)
130 DIM BRKS(10,3),BELTS(10,3),AC(10,3),REP(10,3),
    DESC$(10),INTV(8)
140 DEF DT$(X)=STR$(X-INT(X/100)*100)&"/"&
    STR$(INT(X/100))
150 CALL CLEAR
160 PRINT :"1. Get an existing auto log":
    "2. Create a new auto log"
170 PRINT "3. Save current auto log":
    "4. Routine maint report"
180 PRINT "5. Routine maint update":
    "6. Other repairs report"
190 PRINT "7. Other repairs update":"8. END": : :
200 INPUT "YOUR CHOICE->":A
210 PRINT
```

Listing 4-1—cont. Auto Maintenance Program

```
220 IF A<>8 THEN 240
230 STOP
240 IF (A>0)*(A<8)THEN 270
250 CALL SOUND(200,131,2,262,4,-2,3)
260 GOTO 160
270 ON A GOSUB 1000,3000,5000,7000,9000,11000,13000
280 GOTO 160
1000 REM GET EXISTING LOG
1010 INPUT "ENTER LOG FILE NAME->":LOGF$
1020 OPEN #10:LOGF$,INPUT ,INTERNAL
1030 INPUT #10:AUTONAME$,CURMO,CURYR
1040 FOR I=1 TO 8
1050 INPUT #10:INTV(I)
1060 NEXT I
1070 FOR I=1 TO 10
1080 FOR J=1 TO 3
1090 INPUT #10:OIL(I,J),TUP(I,J),TIRE(I,J)
1100 INPUT #10:BRKS(I,J),BELTS(I,J),AC(I,J),REP(I,J)
1110 NEXT J
1120 INPUT #10:DESC$(I)
1130 NEXT I
1140 CLOSE #10
1150 PRINT : : :AUTONAME$;" LOG READ."
1160 PRINT :"LAST ACCESSED "&STR$(CURMO)&"/"&
     STR$(CURYR): :
1170 GOSUB 19000
1180 RETURN
3000 REM CREATE NEW LOG
3010 IF NNEW=0 THEN 3030
3020 PRINT :C$;AUTONAME$;E$
3030 INPUT "AUTO NAME(NULL TO END)->":Y$
3040 IF Y$="" THEN 3500
3050 AUTONAME$=Y$
3060 PRINT :"CONSULT YOUR AUTO MANUAL TO":
     "FIND THE RECOMMENDED"
3070 PRINT "MAINTENANCE INTERVALS AS":"FOLLOWS:": :
3080 IF NNEW=0 THEN 3100
3090 PRINT :C$;INTV(1);E$
3100 INPUT "OIL CHANGE IN MILES->":INTV(1)
3110 IF NNEW=0 THEN 3130
3120 PRINT :C$;INTV(2);E$
3130 INPUT "OIL CHANGE IN MONTHS->":INTV(2)
3140 IF NNEW=0 THEN 3160
3150 PRINT :C$;INTV(3);E$
3160 INPUT "TUNE UP IN MILES->":INTV(3)
3170 IF NNEW=0 THEN 3190
3180 PRINT :C$;INTV(4);E$
3190 INPUT "TUNE UP IN MONTHS->":INTV(4)
3200 IF NNEW=0 THEN 3220
3210 PRINT :C$;INTV(5);E$
3220 INPUT "TIRE ROTATION IN MILES->":INTV(5)
3230 IF NNEW=0 THEN 3250
3240 PRINT :C$;INTV(6);E$
3250 PRINT "BRAKE REPLACEMENT"
3260 INPUT "     IN MILES->":INTV(6)
3270 IF NNEW=0 THEN 3290
3280 PRINT :C$;INTV(7);E$
3290 PRINT "AIR CONDITIONER MAINTENANCE"
```

Listing 4-1—cont. Auto Maintenance Program

```
3300 INPUT "     IN MONTHS->":INTV(7)
3310 IF NNEW=0 THEN 3330
3320 PRINT :C$;INTV(8);E$
3330 PRINT "BELT CHECK INTERVAL"
3340 INPUT "     IN MONTHS->":INTV(8)
3350 IF NNEW THEN 3500
3360 FOR I=1 TO 10
3370 FOR J=1 TO 3
3380 OIL(I,J)=0
3390 TUP(I,J)=0
3400 TIRE(I,J)=0
3410 BRKS(I,J)=0
3420 BELTS(I,J)=0
3430 AC(I,J)=0
3440 REP(I,J)=0
3450 NEXT J
3460 DESC$(I)=""
3470 NEXT I
3480 GOSUB 19000
3490 RETURN
5000 REM SAVE CURRENT LOG
5010 INPUT "ENTER LOG FILE NAME->":LOGF$
5020 OPEN #20:LOGF$,OUTPUT,INTERNAL
5030 PRINT #20:AUTONAME$,CURMO,CURYR
5040 FOR I=1 TO 8
5050 PRINT #20:INTV(I)
5060 NEXT I
5070 FOR I=1 TO 10
5080 FOR J=1 TO 3
5090 PRINT #20:OIL(I,J),TUP(I,J),TIRE(I,J)
5100 PRINT #20:BRKS(I,J),BELTS(I,J),AC(I,J),REP(I,J)
5110 NEXT J
5120 PRINT #20:DESC$(I)
5130 NEXT I
5140 CLOSE #20
5150 PRINT : : :AUTONAME$;" LOG SAVED.": :
5160 RETURN
7000 REM ROUTINE REPORTS
7010 PRINT : :"1. Service now":"2. Service history":
     "3. END": :
7020 INPUT "YOUR CHOICE->":A
7030 IF A<>3 THEN 7050
7040 RETURN
7050 IF (A>0)*(A<3)THEN 7080
7060 CALL SOUND(200,131,2,262,4,-2,3)
7070 GOTO 7000
7080 ON A GOSUB 7100,7430
7090 GOTO 7000
7100 PRINT : : :TAB(5);"MAINTENANCE REPORT":TAB(8);
     "SERVICE NOW": :
7110 MILES=OIL(1,1)
7120 INM=1
7130 LN=1
7140 GOSUB 16000
7150 DATE=OIL(1,2)
7160 INM=2
7170 GOSUB 15000
7180 MILES=TUP(1,1)
```

Listing 4-1—cont. Auto Maintenance Program

```
7190 INM=3
7200 LN=2
7210 GOSUB 16000
7220 DATE=TUP(1,2)
7230 INM=4
7240 GOSUB 15000
7250 MILES=TIRE(1,1)
7260 INM=5
7270 LN=3
7280 GOSUB 16000
7290 MILES=BRKS(1,1)
7300 INM=6
7310 LN=4
7320 GOSUB 16000
7330 DATE=AC(1,2)
7340 INM=7
7350 LN=5
7360 GOSUB 15000
7370 DATE=BELTS(1,2)
7380 INM=8
7390 LN=6
7400 GOSUB 15000
7410 GOSUB 18000
7420 RETURN
7430 REM SERVICE HISTORY
7440 PRINT : :TAB(7);"SERVICE HISTORY": :
7450 PRINT "1. Oil change":"2. Tune up":
     "3. Tire rotation"
7460 PRINT "4. Brake replacement":"5. Fan belts":
     "6. Air conditioning"
7470 PRINT "7. END": :
7480 INPUT "YOUR CHOICE->":A
7490 LC=1
7500 IF A<>7 THEN 7520
7510 RETURN
7520 IF (A>0)*(A<7)THEN 7550
7530 CALL SOUND(200,131,2,262,4,-2,3)
7540 GOTO 7430
7550 ON A GOTO 7570,7660,7750,7840,7930,8020
7560 GOTO 7430
7570 FOR I=1 TO 10
7580 MILES=OIL(I,1)
7590 DATE=OIL(I,2)
7600 COST=OIL(I,3)
7610 IF MILES=0 THEN 7430
7620 LN=1
7630 GOSUB 8110
7640 NEXT I
7650 GOTO 7430
7660 FOR I=1 TO 10
7670 MILES=TUP(I,1)
7680 DATE=TUP(I,2)
7690 COST=TUP(I,3)
7700 IF MILES=0 THEN 7430
7710 LN=2
7720 GOSUB 8110
7730 NEXT I
7740 GOTO 7430
```

Listing 4-1—cont. Auto Maintenance Program

```
7750 FOR I=1 TO 10
7760 MILES=TIRE(I,1)
7770 DATE=TIRE(I,2)
7780 COST=TIRE(I,3)
7790 IF MILES=0 THEN 7430
7800 LN=3
7810 GOSUB 8110
7820 NEXT I
7830 GOTO 7430
7840 FOR I=1 TO 10
7850 MILES=BRKS(I,1)
7860 DATE=BRKS(I,2)
7870 COST=BRKS(I,3)
7880 LN=4
7890 IF MILES=0 THEN 7430
7900 GOSUB 8110
7910 NEXT I
7920 GOTO 7430
7930 FOR I=1 TO 10
7940 MILES=BELTS(I,1)
7950 DATE=BELTS(I,2)
7960 COST=BELTS(I,3)
7970 IF MILES=0 THEN 7430
7980 LN=5
7990 GOSUB 8110
8000 NEXT I
8010 GOTO 7430
8020 FOR I=1 TO 10
8030 MILES=AC(I,1)
8040 DATE=AC(I,2)
8050 COST=AC(I,3)
8060 IF MILES=0 THEN 7430
8070 LN=6
8080 GOSUB 8110
8090 NEXT I
8100 GOTO 7430
8110 REM PRINT HISTORY
8120 IF LC<>1 THEN 8150
8130 PRINT : : :TAB(6);"SERVICE HISTORY":
     TAB((27-LEN(LAB$(LN)))/2);LAB$(LN)
8140 PRINT :"    DATE    MILEAGE  COST"
8150 PRINT STR$(I);TAB(5);DT$(DATE);TAB(12);MILES;
     TAB(21);COST
8160 LC=LC+1
8170 IF LC<20 THEN 8200
8180 GOSUB 18000
8190 LC=1
8200 RETURN
9000 REM ROUTINE UPDATE
9010 PRINT : : :TAB(7);"ROUTINE UPDATE": :
9020 PRINT "1. Oil change":"2. Tune up":
     "3. Tire rotation"
9030 PRINT "4. Brake replacement":"5. Fan belts":
     "6. Air conditioning"
9040 PRINT "7. Maint intervals":"8. END": :
9050 INPUT "YOUR CHOICE->":A
9060 IF A<>8 THEN 9080
9070 RETURN
```

Listing 4-1—cont. Auto Maintenance Program

```
9080 IF (A>0)*(A<8)THEN 9110
9090 CALL SOUND(200,131,2,262,4,-2,3)
9100 GOTO 9000
9110 ON A GOTO 9140,9280,9420,9560,9700,9840,9980
9120 PRINT : :"NO ACTION TAKEN."
9130 GOTO 9000
9140 PRINT : :LAB$(1): :
9150 GOSUB 10030
9160 IF COST<0 THEN 9120
9170 FOR I=9 TO 1 STEP -1
9180 IF OIL(I,2)=0 THEN 9230
9190 J=I+1
9200 OIL(J,1)=OIL(I,1)
9210 OIL(J,2)=OIL(I,2)
9220 OIL(J,3)=OIL(I,3)
9230 NEXT I
9240 OIL(1,1)=MILES
9250 OIL(1,2)=YR*100+MONTH
9260 OIL(1,3)=COST
9270 GOTO 9000
9280 PRINT : :LAB$(2): :
9290 GOSUB 10030
9300 IF COST<0 THEN 9120
9310 FOR I=9 TO 1 STEP -1
9320 IF TUP(I,2)=0 THEN 9370
9330 J=I+1
9340 TUP(J,1)=TUP(I,1)
9350 TUP(J,2)=TUP(I,2)
9360 TUP(J,3)=TUP(I,3)
9370 NEXT I
9380 TUP(1,1)=MILES
9390 TUP(1,2)=YR*100+MONTH
9400 TUP(1,3)=COST
9410 GOTO 9000
9420 PRINT : :LAB$(3): :
9430 GOSUB 10030
9440 IF COST<0 THEN 9120
9450 FOR I=9 TO 1 STEP -1
9460 IF TIRE(I,2)=0 THEN 9510
9470 J=I+1
9480 TIRE(J,1)=TIRE(I,1)
9490 TIRE(J,2)=TIRE(I,2)
9500 TIRE(J,3)=TIRE(I,3)
9510 NEXT I
9520 TIRE(1,1)=MILES
9530 TIRE(1,2)=YR*100+MONTH
9540 TIRE(1,3)=COST
9550 GOTO 9000
9560 PRINT : :LAB$(4): :
9570 GOSUB 10030
9580 IF COST<0 THEN 9120
9590 FOR I=9 TO 1 STEP -1
9600 IF BRKS(I,2)=0 THEN 9650
9610 J=I+1
9620 BRKS(J,1)=BRKS(I,1)
9630 BRKS(J,2)=BRKS(I,2)
9640 BRKS(J,3)=BRKS(I,3)
9650 NEXT I
```

Listing 4-1—cont. Auto Maintenance Program

```
9660 BRKS(1,1)=MILES
9670 BRKS(1,2)=YR*100+MONTH
9680 BRKS(1,3)=COST
9690 GOTO 9000
9700 PRINT : :LAB$(5): :
9710 GOSUB 10030
9720 IF COST<0 THEN 9120
9730 FOR I=9 TO 1 STEP -1
9740 IF BELTS(I,2)=0 THEN 9790
9750 J=I+1
9760 BELTS(J,1)=BELTS(I,1)
9770 BELTS(J,2)=BELTS(I,2)
9780 BELTS(J,3)=BELTS(I,3)
9790 NEXT I
9800 BELTS(1,1)=MILES
9810 BELTS(1,2)=YR*100+MONTH
9820 BELTS(1,3)=COST
9830 GOTO 9000
9840 PRINT : :LAB$(6): :
9850 GOSUB 10030
9860 IF COST<0 THEN 9120
9870 FOR I=9 TO 1 STEP -1
9880 IF AC(I,2)=0 THEN 9930
9890 J=I+1
9900 AC(J,1)=AC(I,1)
9910 AC(J,2)=AC(I,2)
9920 AC(J,3)=AC(I,3)
9930 NEXT I
9940 AC(1,1)=MILES
9950 AC(1,2)=YR*100+MONTH
9960 AC(1,3)=COST
9970 GOTO 9000
9980 REM INTERVALS
9990 NNEW=-1
10000 GOSUB 3000
10010 NNEW=0
10020 GOTO 9000
10030 REM GET CURRENT DATA
10040 MONTH=CURMO
10050 YR=CURYR
10060 INPUT "USE CURRENT MO/YR(Y/N)":Y$
10070 IF (Y$="Y")+(Y$="y")THEN 10170
10080 IF (Y$<>"N")*(Y$<>"n")THEN 10060
10090 INPUT "ENTER MONTH->":MONTH
10100 IF (MONTH>0)*(MONTH<13)THEN 10130
10110 CALL SOUND(200,131,2,262,4,-2,3)
10120 GOTO 10090
10130 INPUT "ENTER YEAR->":YR
10140 IF (YR>0)*(YR<100)THEN 10170
10150 CALL SOUND(200,131,2,262,4,-2,3)
10160 GOTO 10130
10170 MILES=CURMILES
10180 INPUT "USE CURRENT MILEAGE(Y/N)":Y$
10190 IF (Y$="Y")+(Y$="y")THEN 10220
10200 IF (Y$<>"N")*(Y$<>"n")THEN 10180
10210 INPUT "ENTER MILEAGE->":MILES
10220 INPUT "ENTER COST(-1 TO VOID)->":COST
10230 RETURN
```

Listing 4-1—cont. Auto Maintenance Program

```
11000 REM OTHER REPAIRS REPORT
11010 LC=0
11020 FOR I=1 TO 10
11030 IF REP(I,2)=0 THEN 11110
11040 PRINT :"REPAIR";I:"DATE:";DT$(REP(I,2)):
      "MILEAGE:";REP(I,1)
11050 PRINT "COST:";REP(I,3):"DESCRIPTION: ";DESC$(I)
11060 LC=LC+1
11070 IF LC<2 THEN 11100
11080 GOSUB 18000
11090 LC=0
11100 NEXT I
11110 GOSUB 18000
11120 RETURN
13000 REM OTHER REPAIRS UPDATE
13010 PRINT : :"OTHER REPAIRS": :
13020 GOSUB 10030
13030 IF COST<0 THEN 13170
13040 FOR I=9 TO 1 STEP -1
13050 IF REP(I,2)=0 THEN 13110
13060 J=I+1
13070 REP(J,1)=REP(I,1)
13080 REP(J,2)=REP(I,2)
13090 REP(J,3)=REP(I,3)
13100 DESC$(J)=DESC$(I)
13110 NEXT I
13120 REP(1,1)=MILES
13130 REP(1,2)=YR*100+MONTH
13140 REP(1,3)=COST
13150 INPUT "ENTER DESCRIPTION->":DESC$(1)
13160 RETURN
13170 PRINT : :"NO ACTION TAKEN."
13180 RETURN
15000 REM CHECK DATE DUE
15010 YR=INT(DATE/100)
15020 MONTH=DATE-YR*88+INTV(INM)
15030 CMO=CURYR*12+CURMO
15040 IF CMO>=MONTH THEN 15130
15050 YR=INT(MONTH/12)
15060 MONTH=MONTH-YR*12
15070 IF MONTH<>0 THEN 15100
15080 MONTH=12
15090 YR=YR-1
15100 DATE=YR*100+MONTH
15110 PRINT :LAB$(LN);" DUE ";DT$(DATE)
15120 RETURN
15130 PRINT :LAB$(LN);" DUE NOW."
15140 RETURN
16000 REM CHECK MILEAGE DUE
16010 MILES=MILES+INTV(INM)
16020 IF CURMILES>=MILES THEN 16050
16030 PRINT :LAB$(LN);" DUE AT";MILES;" MILES."
16040 RETURN
16050 PRINT :LAB$(LN);" DUE NOW."
16060 RETURN
18000 REM DELAY FOR KEY STROKE
18010 PRINT :TAB(4);"HIT ANY KEY TO GO ON"
18020 CALL KEY(0,K,S)
```

Listing 4-1—cont. Auto Maintenance Program

```
18030 IF S=0 THEN 18020
18040 RETURN
19000 REM INIT VARIABLES
19010 RESTORE
19020 FOR I=1 TO 6
19030 READ LAB$(I)
19040 NEXT I
19050 C$="CURRENTLY "
19060 E$=" ENTER"
19070 INPUT "ENTER CURRENT MONTH->":CURMO
19080 IF (CURMO>0)*(CURMO<13)THEN 19110
19090 CALL SOUND(200,131,2,262,4,-2,3)
19100 GOTO 19070
19110 INPUT "ENTER CURRENT YEAR->":CURYR
19120 IF (CURYR>0)*(CURYR<100)THEN 19150
19130 CALL SOUND(200,131,2,262,4,-2,3)
19140 GOTO 19110
19150 INPUT "ENTER CURRENT MILEAGE->":CURMILES
19160 RETURN
19170 DATA OIL CHANGE,TUNE UP,TIRE ROTATION
19180 DATA BRAKE REPLACEMENT,FAN BELTS,AIR CONDTIONING
```

FLOOR COVERING PROGRAM

What it does:

The Floor Covering program was designed to help you choose between different floor coverings. After you enter the dimensions for your room, you enter data for up to 20 different floor coverings. The program then calculates the amount of each covering needed and its cost.

Since rooms are not all square, the program determines flooring needs for:

- rectangular rooms
- L-shaped rooms.

If you have an L-shaped room, the program asks for the room dimensions in two parts: the main room and the L-section. Both parts are treated as rectangles in the calculations.

For each flooring material, you enter the following information:

- Material description (such as blue carpet, oak flooring, or ceramic tile)
- Material price per unit (such as 2.75 or 15.99)
- Unit size (how you buy the material, such as per square yard or in 6-inch tiles).

Since the material's unit size affects its cost and the amount you have to buy to cover a floor, we've provided this menu of common unit sizes to choose from:

1. Square foot
2. Square yard
3. 9-inch tiles
4. 6-inch tiles
5. 3-inch tiles.

For each material entered, you'll see how much material you need to purchase and how much it will cost. If you're considering tiles of some sort, the program also tells you how many you'll be using.

Once you've entered all your material choices, you can choose from this menu of available processes:

1. Change room width
2. Change room length
3. Change material/cost/unit
4. Add material/cost/unit
5. Print the data
6. STOP.

You can also use this program to see how much tile you'd need to cover a countertop or table and find out which is most cost effective.

What it shows you:

The Floor Covering program has to be able to calculate area information for two different room shapes: rectangular and L-shaped. The RECT flag is set to show what type of room you're processing. This shows how you can use common routines to process information for several different shapes.

Since tiles come in various sizes, the program also has to calculate the number of 3-, 6-, and 9-inch tiles needed to cover an area. The routine at lines 9000-9260 does the calculation, depending on the value that you set for TILESIZE. Again, a single routine is used for different calculations.

The program must get sufficient information to determine the amount of material needed to cover an area. The easiest way to do this is through a menu of material units (square foot, 3-inch tiles, etc.).

What you can do:

You may decide that you want to calculate areas for other room shapes (maybe a hexagonal room). Just add the shape to the room shape menu in lines 1020 through 1060 and add code to calculate the new area.

You can get flooring material in any number of different units (maybe 1-inch tiles or 3- by 6-inch tiles). To add more choices to the five currently in the program, simply add the unit description to the data statements at lines 140–150 and change the value for NCHOICES (the first value in DATA statement 140) to reflect the additional size information. Then, put in code to determine how much material is needed to cover the area with the new units. You should include code in sections 4000, 7000, 9000, and 10000 to handle the new size.

You can also format a report showing the different materials, units, and costs and write it to a printer. It's easy to add to the main menu at lines 330 and 390 and add a section to GOSUB that does the printing.

If you want to process more than 20 materials at one time, simply increase the array dimensions in line 130.

Table 4-3. Floor Covering Routines

Lines	Function
100-290	Initialize variables and screen.
300-440	Main processing menu.
1000-1240	Get room shape and size and calculate the room area.
1250-1430	Get the material information (description, unit size, cost) and calculate how much is needed to cover the room area. Print the results to the screen.
2000-2090	Change the room's width and recalculate the area and amount of materials needed.
3000-3090	Change the room's length and recalculate the area and amount of materials needed.
4000-4330	Change material information (description/unit size/cost) and recalculate the cost if necessary.
5000-5050	Add more material information.
6000-6130	Print the flooring requirements for the various materials.
7000-7300	Get material information.
8000-8140	Calculate room area.
9000-9260	Calculate number of tiles needed to cover the room area.
10000-10160	Recalculate costs.
11000-11040	Delay function.

Table 4-4. Floor Covering Variables

Variable	Description
AREA	Room area (length times width) in square feet.
COST(20)	Room cost for a material.
INLENGTH	Room length in inches.
INWIDTH	Room width in inches.
LAREA	Area of L-section in square feet.
LENGTH	Room length in feet.
LINLEN	L-section length in inches.
LINWIDTH	L-section width in inches.
LLENGTH	L-section length in feet.
LSQYDS	Area of L-section in square yards.
LWIDTH	L-section width in feet.
MAT$(20)	Material description.
NTILES(20)	Number of tiles needed for a material.
NUMLTILES	Number of tiles needed to cover the L-section.
NUMTILES	Number of tiles needed to cover the room.
PRICE(20)	Price of a material.
RD	Function that rounds to two decimal places (nn.nn).
RECT	Flag for rectangular (RECT=1) or L-shaped (RECT=2) room.
SIZE$(10)	Unit sizes for materials—used in menus.
SQYDS	Number of square yards in the room area.
TILESIZE	Size of the tile in inches.
TLEN	Number of tiles needed for the room's length.
TLLEN	Number of tiles needed for the L's length.
TLWIDE	Number of tiles needed for the L's width.
TWIDE	Number of tiles needed for the room's width.
UNIT(20)	Unit for a material (chosen from a menu).
WIDTH	Room width in feet.

Listing 4-2. Floor Covering Program

```
100 REM THIS PROGRAM CALCULATES THE COST OF VARIOUS
110 REM FLOORING MATERIALS
120 OPTION BASE 1
130 DIM MAT$(20),PRICE(20),UNIT(20),COST(20),
    SIZE$(10),NTILES(20)
140 DATA 5,"square foot","square yard","9-inch tile",
    "6-inch tile"
150 DATA "3-inch tile"
160 DEF RD(X)=INT(X*100+.5)/100
170 REM INITIALIZE
180 READ NCHOICES
190 FOR I=1 TO NCHOICES
200 READ SIZE$(I)
210 NEXT I
220 REM PRINT INSTRUCTIONS
230 CALL CLEAR
240 PRINT : : :"This program helps you":
    "decide which flooring"
250 PRINT "material to use.": :
    "Enter the room dimensions"
```

Listing 4-2—cont. Floor Covering Program

```
260 PRINT "and prices for various":
    "flooring materials."
270 PRINT :"The program calculates":
    "costs for your choices."
280 PRINT : : : : :
290 GOSUB 11000
300 REM GET ROOM DIMENSIONS AND MATERIAL INFORMATION
310 GOSUB 1000
320 REM MAIN PROCESSING
330 PRINT :"You can:": :" 1 Change room width":
    " 2 Change room length"
340 PRINT " 3 Change material/cost/unit":
    " 4 Add material/cost/unit"
350 PRINT " 5 Print your data":" 6 STOP"
360 PRINT
370 INPUT "Your choice -> ":ANS
380 PRINT
390 IF (ANS<1)+(ANS>6)THEN 330
400 IF ANS<6 THEN 420
410 STOP
420 PRT=0
430 ON ANS GOSUB 2000,3000,4000,5000,6000
440 GOTO 330
1000 REM GET ROOM SHAPE AND SIZE
1010 DONE=0
1020 PRINT "You can calculate costs for:":
     " 1 a rectangular room"
1030 PRINT " 2 an L-shaped room"
1040 PRINT
1050 INPUT "Enter room type-> ":RECT
1060 IF (RECT<1)+(RECT>2)THEN 1000
1070 IF RECT=1 THEN 1100
1080 PRINT :"You have an L-shaped room.":
     "Measure the room as two":"rectangles."
1090 PRINT "Enter the larger rectangle":"first."
1100 PRINT : :"Enter room size in feet."
1110 INPUT "Room width is -> ":WIDTH
1120 IF WIDTH<=0 THEN 1110
1130 INPUT "Room length is-> ":LENGTH
1140 IF LENGTH<=0 THEN 1130
1150 IF RECT=1 THEN 1210
1160 INPUT "Room's L-width is -> ":LWIDTH
1170 IF LWIDTH<=0 THEN 1160
1180 INPUT "Room's L-length is-> ":LLENGTH
1190 IF LLENGTH<=0 THEN 1180
1200 REM CALCULATE ROOM SIZE
1210 GOSUB 8000
1220 PRINT :"Your room is ":AREA;" sq ft":SQYDS;
     " sq yds"
1230 PRINT :"Enter your material choices,":
     "price per unit,"
1240 PRINT "and unit size":"Enter DONE to finish."
1250 N=0
1260 N=N+1
1270 REM GET MATERIAL DATA
1280 GOSUB 7000
1290 IF DONE=0 THEN 1260
1300 REM DONE GETTING DATA, PRINT DATA
```

Listing 4-2—cont. Floor Covering Program

```
1310 PRINT :"Room is";LENGTH;" by ";WIDTH:AREA;
     " sq ft,":SQYDS;" sq yds"
1320 IF RECT=1 THEN 1340
1330 PRINT "L is";LLENGTH;" by ";LWIDTH:LAREA;
     " sq ft,":LSQYDS;" sq yds"
1340 PRINT :"Costs are:"
1350 PRINT
1360 FOR I=1 TO N
1370 PRINT I;MAT$(I);TAB(18);"$";STR$(COST(I))
1380 IF (UNIT(I)<3)THEN 6110
1390 PRINT TAB(10);NTILES(I);SIZE$(UNIT(I))
1400 NEXT I
1410 PRINT
1420 GOSUB 11000
1430 RETURN
2000 REM CHANGE THE ROOM WIDTH
2010 PRINT :"Room width is ";WIDTH;" feet"
2020 INPUT "Enter new width in feet -> ":WIDTH
2030 IF RECT=1 THEN 2070
2040 PRINT :"L width is ";LWIDTH;" feet"
2050 INPUT "Enter new width in feet -> ":LWIDTH
2060 REM REDO ROOM SIZE AND COSTS FOR MATERIALS
2070 GOSUB 8000
2080 GOSUB 10000
2090 RETURN
3000 REM CHANGE THE ROOM LENGTH
3010 PRINT :"Room length is ";LENGTH;" feet"
3020 INPUT "Enter new length in feet -> ":LENGTH
3030 IF RECT=1 THEN 3070
3040 PRINT :"L length is ";LLENGTH;" feet"
3050 INPUT "Enter new length in feet -> ":LLENGTH
3060 REM REDO ROOM SIZE AND COSTS FOR MATERIALS
3070 GOSUB 8000
3080 GOSUB 10000
3090 RETURN
4000 REM CHANGE MATERIAL INFORMATION
4010 PRT=0
4020 PRINT "You can change any of ";STR$(N);
     " materials (0 if done)"
4030 PRINT "MATERIAL";TAB(17);"PRICE"
4040 FOR I=1 TO N
4050 PRINT I;MAT$(I);TAB(17);"$";STR$(PRICE(I))
4060 NEXT I
4070 PRINT
4080 INPUT "Change which one -> ":CHOOSE
4090 IF (CHOOSE<0)+(CHOOSE>N)THEN 4000
4100 IF CHOOSE<>0 THEN 4120
4110 RETURN
4120 INPUT "Material -> ":MAT$(CHOOSE)
4130 INPUT "Price ->":PRICE(CHOOSE)
4140 FOR I=1 TO NCHOICES
4150 PRINT I;"=";SIZE$(I)
4160 NEXT I
4170 INPUT "Enter unit -> ":ANS
4180 IF (ANS<1)+(ANS>NCHOICES)THEN 4140
4190 UNIT(CHOOSE)=ANS
4200 ON ANS GOTO 4210,4230,4250,4280,4310
4210 COST(CHOOSE)=RD(AREA*PRICE(CHOOSE))
```

Listing 4-2—cont. Floor Covering Program

```
4220 GOTO 4000
4230 COST(CHOOSE)=RD(SQYDS*PRICE(CHOOSE))
4240 GOTO 4000
4250 TILESIZE=9
4260 GOSUB 9000
4270 GOTO 4000
4280 TILESIZE=6
4290 GOSUB 9000
4300 GOTO 4000
4310 TILESIZE=3
4320 GOSUB 9000
4330 GOTO 4000
5000 REM ADD MATERIAL INFORMATION
5010 N=N+1
5020 DONE=0
5030 GOSUB 7000
5040 IF DONE=0 THEN 5000
5050 RETURN
6000 REM PRINT THE DATA
6010 CALL CLEAR
6020 PRINT "Room is";LENGTH;" by ";WIDTH:AREA;
     " sq ft,":SQYDS;" sq yds"
6030 IF RECT=1 THEN 6050
6040 PRINT "L is";LLENGTH;" by ";LWIDTH:LAREA;
     " sq ft,":LSQYDS;" sq yds"
6050 PRINT :"Costs are:"
6060 PRINT
6070 FOR I=1 TO N
6080 PRINT MAT$(I);TAB(18);"$";STR$(COST(I))
6090 IF (UNIT(I)<3)THEN 6110
6100 PRINT TAB(10);NTILES(I);SIZE$(UNIT(I))
6110 NEXT I
6120 PRINT
6130 RETURN
7000 REM GET MATERIAL NAME/PRICE/UNIT SIZE
7010 PRT=1
7020 INPUT "Material -> ":MAT$(N)
7030 IF MAT$(N)="DONE" THEN 7280
7040 INPUT "Price ->":PRICE(N)
7050 PRINT "Material unit price can be:"
7060 FOR I=1 TO NCHOICES
7070 PRINT I;TAB(4);SIZE$(I)
7080 NEXT I
7090 INPUT "Your choice -> ":ANS
7100 IF (ANS<1)+(ANS>NCHOICES)THEN 7050
7110 UNIT(N)=ANS
7120 ON ANS GOTO 7130,7150,7170,7200,7230
7130 COST(N)=RD(AREA*PRICE(N))
7140 GOTO 7260
7150 COST(N)=RD(SQYDS*PRICE(N))
7160 GOTO 7260
7170 TILESIZE=9
7180 GOSUB 9000
7190 GOTO 7260
7200 TILESIZE=6
7210 GOSUB 9000
7220 GOTO 7260
7230 TILESIZE=3
```

Listing 4-2—cont. Floor Covering Program

```
7240 GOSUB 9000
7250 GOTO 7260
7260 PRINT "Room cost is: $";STR$(COST(N))
7270 RETURN
7280 N=N-1
7290 DONE=1
7300 RETURN
8000 REM CALC ROOM SIZE
8010 INWIDTH=12*WIDTH
8020 INLENGTH=12*LENGTH
8030 AREA=WIDTH*LENGTH
8040 SQYDS=AREA/9
8050 IF RECT=1 THEN 8120
8060 LAREA=RD(LWIDTH*LLENGTH)
8070 LSQYDS=RD(LAREA/9)
8080 LINWIDTH=12*LWIDTH
8090 LINLEN=12*LLENGTH
8100 AREA=AREA+LAREA
8110 SQYDS=SQYDS+LSQYDS
8120 AREA=RD(AREA)
8130 SQYDS=RD(SQYDS)
8140 RETURN
9000 REM CALCULATE NUMBER OF TILES NEEDED
9010 TWIDE=INWIDTH/TILESIZE
9020 IF TWIDE=INT(TWIDE)THEN 9040
9030 TWIDE=TWIDE+1
9040 TWIDE=INT(TWIDE)
9050 TLEN=INLENGTH/TILESIZE
9060 IF TLEN=INT(TLEN)THEN 9080
9070 TLEN=TLEN+1
9080 TLEN=INT(TLEN)
9090 NUMTILES=TLEN*TWIDE
9100 REM IF RECT ROOM, THEN DONE
9110 IF RECT=1 THEN 9220
9120 TLWIDE=LINWIDTH/TILESIZE
9130 IF TLWIDE=INT(TLWIDE)THEN 9150
9140 TLWIDE=TLWIDE+1
9150 TLWIDE=INT(TLWIDE)
9160 TLLEN=LINLEN/TILESIZE
9170 IF TLLEN=INT(TLLEN)THEN 9190
9180 TLLEN=TLLEN+1
9190 TLLEN=INT(TLLEN)
9200 NUMLTILES=TLLEN*TLWIDE
9210 NUMTILES=NUMTILES+NUMLTILES
9220 IF PRT=0 THEN 9240
9230 PRINT "You need ";STR$(NUMTILES);"  ";
     STR$(TILESIZE);"-in tiles."
9240 COST(N)=RD(NUMTILES*PRICE(N))
9250 NTILES(N)=NUMTILES
9260 RETURN
10000 REM RECALC COSTS
10010 MAXNUM=N
10020 FOR N=1 TO MAXNUM
10030 ON UNIT(N)GOTO 10040,10060,10080,10100,10120
10040 COST(N)=RD(AREA*PRICE(N))
10050 GOTO 10140
10060 COST(N)=RD(SQYDS*PRICE(N))
10070 GOTO 10140
```

Listing 4-2—cont. Floor Covering Program

```
10080 TILESIZE=9
10090 GOTO 10130
10100 TILESIZE=6
10110 GOTO 10130
10120 TILESIZE=3
10130 GOSUB 9000
10140 NEXT N
10150 N=MAXNUM
10160 RETURN
11000 REM DELAY
11010 PRINT "Press any key to continue.";
11020 CALL KEY(0,K,S)
11030 IF S=0 THEN 11020
11040 RETURN
```

RECIPE CONVERTER PROGRAM

What it does:

The Recipe Converter program helps you to increase or decrease the ingredients in a recipe. When you have a recipe that serves two and you want to increase it to serve six, it's sometimes inconvenient to convert the amounts.

For example, if you want to increase a recipe by three, you can always just add three of everything in the recipe. But, when you're dealing with 6 teaspoons of something, wouldn't it be easier if you knew you could use 2 Tablespoons of it—a lot less measuring to do.

So, to help all you cooks out there, we've produced a program that will do the following:

1. Convert temperature (Farenheit to centigrade)
2. Convert temperature (centigrade to Fahrenheit)
3. Enter ingredients for a recipe.
4. Increase the amounts for the ingredients in a recipe.
5. Decrease the amounts for the ingredients in a recipe.
6. Print the original recipe.
7. Print the adjusted recipe.

The temperature conversions take into account the fact that most ovens are calibrated in 25 degree increments (for Fahrenheit) and 10 degree increments (for centigrade).

The ingredient conversions are based on these substitutions:

- 8 pinches = 1 teaspoon
- 3 teaspoons = 1 Tablespoon
- 2 Tablespoons = 1 fluid ounce
- 8 fluid ounces = 1 cup
- 2 cups = 1 pint
- 2 pints = 1 quart
- 4 quarts = 1 gallon
- 16 ounces = 1 pound.

When the recipes are printed, the ingredients are shown with the adjusted amounts and the adjustment factor times the original amount. For example: 2 Tbl (3 × 2 tsp) parsley.

You don't have to reenter the ingredients when you want to try different increase/decrease amounts for the same recipe.

What it shows you:

One thing the Recipe Converter program shows you is how to do several conversions until you finally can't adjust an ingredient any further. This is necessary when you want to make a large adjustment (like eight times the original amount).

For example, you're increasing one cup by a factor of eight. You'll need eight cups. But eight cups is also four pints. And, then again, four pints is two quarts. Should you go further and increase to gallons? Stop at quarts? Stay with pints?

The program uses the DONE flag to tell the conversion routine that begins at line 6000 when the conversion can't go any further.

This program also shows you how to use GOSUBs to do rather small amounts of processing. Converting teaspoons to Tablespoons really isn't all that much code. But, by isolating it in a GOSUB routine, you can easily change it or even remove it without much problem.

What you can do:

The program currently processes up to 50 ingredients for a recipe. If you have some really long recipes, you can increase this maximum. Be sure to adjust the arrays for the ingredients and measures also.

If you have a printer, you can write a routine to print the original and adjusted recipes. Just add another choice to the main menu at lines 500–600 and GOSUB to the appropriate routine.

You might want to build a library of recipes on a storage device. If

you have a disk, you can build a direct access file of recipes. On a cassette, you can store each recipe separately or build a file with some special indicators between recipes.

Then, you can store your recipes to your tape/disk and read them later. This is a bit more difficult to do than printing a recipe. But, with a bit of patience, you can do it. Just remember that everything is menu driven. Add to the menu and add GOSUB routines to do the processing.

Table 4-5. Recipe Converter Routines

Lines	Function
100-490	Initialize variables and screen.
500-600	Main processing.
1000-1090	Convert temperature (Fahrenheit to centigrade).
1500-1580	Convert temperature (centigrade to Fahrenheit).
2000-2030	Get ingredients for new recipe.
2500-2690	Increase a recipe (get ingredients if necessary).
3000-3190	Decrease a recipe (get ingredients if necessary).
4000-4070	Determine whether to print the new or adjusted recipe.
4500-4750	Print the new/adjusted recipe.
5000-5220	Get the actual recipe ingredients.
6000-7270	Actually adjust the ingredient amounts in the recipe.
8000-8040	Delay routine.

Table 4-6. Recipe Converter Variables

Variable	Description
ABBR$(10)	Abbreviations for the ingredient measurements.
ADJMEAS	Used in calculating the adjusted amounts.
ADJUST	Adjustment factor for increasing/decreasing the recipe amounts.
AMOUNT	Used in calculating the adjusted amounts.
AMT(50,2)	Amount of each ingredient. AMT(n,1) is the original recipe amount. AMT(n,2) is the adjusted amount.
CEL	Centigrade degrees.
DONE	Flag used in adjusting the amounts. When DONE=0, more adjusting is needed. When DONE=1, the ingredient amount can't be adjusted further.
FAREN	Fahrenheit degrees.
INGRED$(50)	Recipe ingredient descriptions.
MEASURE$ (50,2)	Recipe measure indicator.
NUMING	Number of ingredients in the recipe.
NUMMEAS (50,2)	Code for the ingredient measure (e.g., 2=tsp). NUMMEAS(n,1) is the original measure. NUMMEAS(n,2) is the adjusted measure.
NUNITS	Maximum number of different adjustment units (currently 10).
NW	New serving value.
ORIG	Original serving value.
RD	Rounding function that rounds to two decimal places (nn.nn).
UNIT$(10)	Unit descriptions for the measures (e.g., tsp).

Listing 4-3. Recipe Converter Program

```
100 REM THIS PROGRAM HELPS YOU TO INCREASE/DECREASE
110 REM AMOUNTS FOR RECIPE INGREDIENTS AND CONVERT
120 REM TEMPERATURES BETWEEN FARENHEIT AND CELSIUS
130 REM
140 REM TEMPERATURE CONVERSIONS ARE TO THE NEAREST
150 REM 10 DEGREES (C) OR NEAREST 25 DEGREES (F)
160 REM
170 REM WITH THIS PROGRAM, YOU CAN:
180 REM  1. CONVERT TEMPERATURES (F->C)
190 REM  2. CONVERT TEMPERATURES (C->F)
200 REM  3. ENTER A RECIPE
210 REM  4. INCREASE A RECIPE
220 REM  5. DECREASE A RECIPE
230 REM  6. PRINT THE ORIGINAL RECIPE
240 REM  7. PRINT THE ADJUSTED RECIPE
250 REM
260 OPTION BASE 1
270 DEF RD(X)=INT(X*100+.5)/100
280 DIM INGRED$(50),AMT(50,2),MEASURE$(50,2),
    NUMMEAS(50,2)
290 DIM UNIT$(10),ABBR$(10)
300 REM
```

Listing 4-3—cont. Recipe Converter Program

```
310 REM  INITIALIZATION - READ IN THE TABLE
320 REM  OF MEASUREMENTS AND ABBREVIATIONS
330 REM
340 DATA 10
350 DATA pinches,pinches,teaspoons,tsp,Tablespoons,Tbl
360 DATA fluid ounces,fl oz,cups,C,pints,pt,quarts,qt
370 DATA gallons,gal,"ounces(weight)",oz,pounds,lb
380 READ NUNITS
390 FOR I=1 TO NUNITS
400 READ UNIT$(I),ABBR$(I)
410 NEXT I
420 REM NOW, BEGIN PROCESSING
430 CALL CLEAR
440 PRINT "* Recipe Conversion Helper *": : :
    "This program helps you to:"
450 PRINT :" convert temperatures":
    "     (F to C, C to F)"
460 PRINT :" increase/decrease recipe":
    "     ingredients."
470 PRINT : : :
480 GOSUB 8000
490 CALL CLEAR
500 PRINT "You can:": :" 1 Convert temp (F->C)":
    " 2 Convert temp (C->F)"
510 PRINT " 3 Enter recipe ingredients":
    " 4 Increase recipe"
520 PRINT " 5 Decrease recipe":
    " 6 Print original recipe"
530 PRINT " 7 Print adjusted recipe":" 8 Stop"
540 PRINT
550 INPUT "Your choice -> ":ANS
560 IF (ANS<1)+(ANS>8)THEN 500
570 IF ANS=8 THEN 600
580 ON ANS GOSUB 1000,1500,2000,2500,3000,4000,4040
590 GOTO 500
600 STOP
1000 REM CONVERT TEMP F->C
1010 REM ROUND C TEMP TO NEAREST 10 DEGREES
1020 REM
1030 PRINT :"Enter temperature in"
1040 INPUT "degrees F -> ":FAREN
1050 CEL=((FAREN-32)*5)/9
1060 CEL=INT((CEL+5)/10)*10
1070 PRINT :FAREN;" degrees F =":CEL;" degrees C": :
1080 GOSUB 8000
1090 RETURN
1500 REM CONVERT TEMP C->F
1510 REM ROUND F TEMP TO NEAREST 25 DEGREES
1520 PRINT :"Enter temperature in"
1530 INPUT "degrees C ->":CEL
1540 FAREN=32+((CEL*9)/5)
1550 FAREN=INT((FAREN+12.5)/25)*25
1560 PRINT :CEL;" degrees C =":FAREN;" degrees F": :
1570 GOSUB 8000
1580 RETURN
2000 REM GET NEW RECIPE
2010 NUMING=0
```

Listing 4-3—cont. Recipe Converter Program

```
2020 GOSUB 5000
2030 RETURN
2500 REM INCREASE A RECIPE
2510 REM FIRST, GET INGREDIENTS
2520 IF NUMING=0 THEN 2580
2530 INPUT "Enter a new recipe (Y/N) -> ":Y$
2540 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 2570
2550 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 2590
2560 GOTO 2530
2570 NUMING=0
2580 GOSUB 5000
2590 PRINT
2600 INPUT "This recipe serves -> ":ORIG
2610 IF ORIG<0 THEN 2590
2620 PRINT
2630 INPUT "Increase to serve -> ":NW
2640 IF NW<0 THEN 2630
2650 PRINT
2660 ADJUST=NW/ORIG
2670 REM ADJUST THE AMOUNTS
2680 GOSUB 6000
2690 RETURN
3000 REM DECREASE A RECIPE
3010 REM FIRST, GET THE INGREDIENTS
3020 IF NUMING=0 THEN 3080
3030 INPUT "Enter a new recipe (Y/N) -> ":Y$
3040 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 3070
3050 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 3090
3060 GOTO 3030
3070 NUMING=0
3080 GOSUB 5000
3090 PRINT
3100 INPUT "This recipe serves -> ":ORIG
3110 IF ORIG<0 THEN 3090
3120 PRINT
3130 INPUT "Decrease to serve -> ":NW
3140 IF NW<0 THEN 3130
3150 PRINT
3160 ADJUST=NW/ORIG
3170 REM ADJUST THE AMOUNTS
3180 GOSUB 6000
3190 RETURN
4000 REM PRINT ORIGINAL RECIPE
4010 PRT=1
4020 GOSUB 4500
4030 RETURN
4040 REM PRINT NEW RECIPE
4050 PRT=2
4060 GOSUB 4500
4070 RETURN
4500 REM PRINT THE RECIPE
4510 IF NUMING>0 THEN 4540
4520 PRINT :"You haven't entered any":
     "recipe data yet.": :
4530 RETURN
4540 GOSUB 4690
4550 FOR I=1 TO NUMING
```

Listing 4-3—cont. Recipe Converter Program

```
4560 TMP$=" "
4570 IF PRT=1 THEN 4600
4580 PRTADJ=RD(ADJUST*AMT(I,1))
4590 TMP$=" ("&STR$(PRTADJ)&" "&MEASURE$(I,1)&") "
4600 PRINT STR$(AMT(I,PRT));" ";MEASURE$(I,PRT);
     TMP$;INGRED$(I)
4610 LINES=LINES+1
4620 IF LINES<20 THEN 4650
4630 GOSUB 8000
4640 GOSUB 4690
4650 NEXT I
4660 PRINT
4670 GOSUB 8000
4680 RETURN
4690 TMP$="Recipe for "
4700 IF PRT=1 THEN 4720
4710 TMP$="Adjusted recipe for "
4720 PRINT :TMP$;NAME$
4730 PRINT
4740 LINES=2
4750 RETURN
5000 REM GET THE INGREDIENTS FOR THE RECIPE
5010 INPUT "Enter the recipe name -> ":NAME$
5020 PRINT
5030 NUMING=NUMING+1
5040 PRINT :"Enter each ingredient"
5050 INPUT " (DONE when finished) -> ":INGRED$(NUMING)
5060 IF INGRED$(NUMING)="DONE" THEN 5210
5070 PRINT "Choose from these units:"
5080 FOR I=1 TO NUNITS
5090 PRINT STR$(I);TAB(5);UNIT$(I)
5100 NEXT I
5110 PRINT
5120 INPUT "Your unit -> ":ANS
5130 IF (ANS<1)+(ANS>NUNITS)THEN 5070
5140 MEASURE$(NUMING,1)=ABBR$(ANS)
5150 NUMMEAS(NUMING,1)=ANS
5160 MSG$="How many "&ABBR$(ANS)&" -> "
5170 INPUT MSG$:ANS
5180 IF ANS<0 THEN 5140
5190 AMT(NUMING,1)=ANS
5200 GOTO 5030
5210 NUMING=NUMING-1
5220 RETURN
6000 REM ADJUST THE RECIPE
6010 FOR I=1 TO NUMING
6020 DONE=0
6030 ADJMEAS=NUMMEAS(I,1)
6040 AMOUNT=AMT(I,1)*ADJUST
6050 ON ADJMEAS GOSUB 6120,6200,6290,6440,6590,6740,
     6890,7040,7130,7210
6060 IF DONE=0 THEN 6050
6070 AMT(I,2)=RD(AMOUNT)
6080 NUMMEAS(I,2)=ADJMEAS
6090 MEASURE$(I,2)=ABBR$(ADJMEAS)
6100 NEXT I
6110 RETURN
6120 REM CONVERT PINCHES
```

Listing 4-3—cont. Recipe Converter Program

```
6130 REM 8 PINCHES = 1 TEASPOON
6140 IF ADJUST<0 THEN 6180
6150 AMOUNT=AMOUNT/8
6160 ADJMEAS=ADJMEAS+1
6170 IF AMOUNT>=1 THEN 6190
6180 DONE=1
6190 RETURN
6200 REM CONVERT TEASPOONS
6210 REM 3 TEASPOONS = 1 TABLESPOON
6220 IF AMOUNT<3 THEN 6270
6230 REM INCREASE UNITS
6240 AMOUNT=AMOUNT/3
6250 ADJMEAS=ADJMEAS+1
6260 RETURN
6270 DONE=1
6280 RETURN
6290 REM CONVERT TABLESPOONS
6300 IF AMOUNT<2 THEN 6360
6310 REM INCREASE UNITS
6320 REM 2 TABLESPOONS = 1 FL OZ
6330 AMOUNT=AMOUNT/2
6340 ADJMEAS=ADJMEAS+1
6350 RETURN
6360 IF AMOUNT>=1 THEN 6420
6370 REM DECREASE UNITS
6380 REM 1 TABLESPOON = 3 TEASPOONS
6390 AMOUNT=AMOUNT*3
6400 ADJMEAS=ADJMEAS-1
6410 RETURN
6420 DONE=1
6430 RETURN
6440 REM CONVERT FLUID OUNCES
6450 IF AMOUNT<8 THEN 6510
6460 REM INCREASE UNITS
6470 REM 8 FL OZ = 1 CUP
6480 AMOUNT=AMOUNT/8
6490 ADJMEAS=ADJMEAS+1
6500 RETURN
6510 IF AMOUNT>=1 THEN 6570
6520 REM DECREASE UNITS
6530 REM 1 FL OZ = 2 TABLESPOONS
6540 AMOUNT=AMOUNT*2
6550 ADJMEAS=ADJMEAS-1
6560 RETURN
6570 DONE=1
6580 RETURN
6590 REM CONVERT CUPS
6600 IF AMOUNT<2 THEN 6660
6610 REM INCREASE UNITS
6620 REM 1 PINT = 2 CUPS
6630 AMOUNT=AMOUNT/2
6640 ADJMEAS=ADJMEAS+1
6650 RETURN
6660 IF AMOUNT>=1 THEN 6720
6670 REM DECREASE UNITS
6680 REM 8 FL OZ = 1 CUP
6690 AMOUNT=AMOUNT*8
6700 ADJMEAS=ADJMEAS-1
```

Listing 4-3—cont. Recipe Converter Program

```
6710 RETURN
6720 DONE=1
6730 RETURN
6740 REM CONVERT PINTS
6750 REM 2 PINTS = 1 QUART
6760 IF AMOUNT<2 THEN 6810
6770 REM INCREASE UNITS
6780 AMOUNT=AMOUNT/2
6790 ADJMEAS=ADJMEAS+1
6800 RETURN
6810 IF AMOUNT>=1 THEN 6870
6820 REM DECREASE UNITS
6830 REM 1 PINT = 2 CUPS
6840 AMOUNT=AMOUNT*2
6850 ADJMEAS=ADJMEAS-1
6860 RETURN
6870 DONE=1
6880 RETURN
6890 REM CONVERT QUARTS
6900 IF AMOUNT<4 THEN 6960
6910 REM INCREASE UNITS
6920 REM 1 GALLON = 4 QUARTS
6930 AMOUNT=AMOUNT/4
6940 ADJMEAS=ADJMEAS+1
6950 RETURN
6960 IF AMOUNT>=1 THEN 7020
6970 REM DECREASE UNITS
6980 REM 1 QUART = 2 PINTS
6990 AMOUNT=AMOUNT*2
7000 ADJMEAS=ADJMEAS-1
7010 RETURN
7020 DONE=1
7030 RETURN
7040 REM CONVERT GALLONS
7050 IF AMOUNT<1 THEN 7080
7060 DONE=1
7070 RETURN
7080 REM DECREASE UNITS
7090 REM 1 GALLON = 4 QUARTS
7100 AMOUNT=AMOUNT*4
7110 ADJMEAS=ADJMEAS-1
7120 RETURN
7130 REM CONVERT OUNCES
7140 REM 16 OUNCES = 1 POUND
7150 IF AMOUNT<16 THEN 7190
7160 REM INCREASE UNITS
7170 AMOUNT=AMOUNT/16
7180 ADJMEAS=ADJMEAS+1
7190 DONE=1
7200 RETURN
7210 REM CONVERT POUNDS
7220 IF AMOUNT>=(1/16)THEN 7260
7230 REM DECREASE UNITS
7240 AMOUNT=AMOUNT*16
7250 ADJMEAS=ADJMEAS-1
7260 DONE=1
7270 RETURN
8000 REM PAUSE
```

Listing 4-3—cont. Recipe Converter Program

```
8010 PRINT "Press any key to continue."
8020 CALL KEY(0,K,S)
8030 IF S=0 THEN 8020
8040 RETURN
```

WALLPAPER/PAINT PROGRAM

What it does:

The Wallpaper/Paint program determines how much material you'll need to paper/paint an area.

Since wallpaper comes in several different size rolls, the program determines how many single/double/triple rolls you'll need. One single roll covers 30 square feet.

Paint has different covering capacities. The program gives you the number of gallons needed for 100/200/300/400 square feet per gallon coverage.

The algorithm used to calculate the amount of material needed lets you enter the number of doors and/or windows on each wall. If your windows aren't oversized, you don't need to enter any data about them. However, if your home has very large windows and doors, you should enter the square footage for each one.

What it shows you:

The Wallpaper/Paint program shows you how to get dependent information. That is, first you get the number of walls you're going to process. Then, for each wall, you get the number of doors and windows in the wall. Finally, you get the sizes for each door and window. All data for a wall is collected before beginning to process the next wall.

Each time data is requested, it's checked for reasonableness. That means you can't exceed the number of doors/windows/walls that the program can process and you can't say that you have a negative number of doors.

The DELAY routine makes it easy for you to read the screen before the information scrolls off the top.

What you can do:

You can add an algorithm to determine how much paper is needed to match patterns. At this time, the program simply reminds you to buy extra paper if your selected pattern has a large repeat.

You might want to include a section that calculates costs for var-

ious papers/paints/combinations. Just add more menu choices and appropriate routines.

Since some rooms have walls with different height ceilings (like rooms with cathedral ceilings), you can include statements to get the ceiling height (or average ceiling height) for each wall. Remember to change the algorithm that calculates the wall area if you change the ceiling heights.

Table 4-7. Wallpaper/Paint Routines

Lines	Function
100-240	Initialize variables and screen.
250-570	Main processing.
1000-1060	Get the room height.
2000-2530	Get the information for each wall: the wall length, the number of doors and windows, and the sizes of the doors and windows.
3000-3240	Calculate the wall area. Remove the door and window areas.
4000-4150	Calculate the number of gallons of paint and rolls of wallpaper needed to cover the wall area.
5000-5170	Print your wallpaper/paint report.
6000-6040	Delay routine.

Table 4-8. Wallpaper/Paint Variables

Variable	Description
CEILING	Ceiling height.
DOORAREA	Total area of all the doors in the room.
DOORS(10)	Number of doors on each wall.
DOORSIZE (10,4)	Size, in square feet, of each door.
LENGTH	Length of a wall.
MAXDOOR	Maximum number of doors you can have on each wall (currently 4).
MAXWALL	Maximum number of walls you can process (currently 10).
MAXWIN	Maximum number of windows you can have on each wall (currently 6).
NUMWALLS	Number of walls in the room.
PAINT(4)	Number of gallons of paint needed to cover the wall area at 100/200/300/400 square feet/gallon.
PAPER(3)	Number of rolls of wallpaper needed to cover the wall area at 30/60/90 square feet per roll.
ROOMAREA	Total wall area (Wall Area − DOORAREA − WINAREA).
WALLS(10)	Length of each wall.
WINAREA	Total area of all the windows in the room.
WINDOWS(10)	Number of windows on each wall.
WINSIZE(10,6)	Size, in square feet, of each window.

Listing 4-4. Wallpaper/Paint Program

```
100 REM THIS PROGRAM HELPS YOU TO DECIDE HOW MUCH
    WALLPAPER
110 REM OR PAINT YOU NEED FOR A ROOM
120 REM
130 OPTION BASE 1
140 DIM WALLS(10),DOORS(10),DOORSIZE(10,4)
150 DIM WINDOWS(10),WINSIZE(10,6)
160 DIM PAINT(4),PAPER(3),ROLL$(3)
170 DEF RD(X)=INT(X*100+.5)/100
180 MAXDOOR=4
190 MAXWIN=6
200 MAXWALL=10
210 ROLL$(1)="single"
220 ROLL$(2)="double"
230 ROLL$(3)="triple"
240 CALL CLEAR
250 PRINT "**** Decorating Helper ****"
260 PRINT : :"This program helps you find":
    "out how much paint or"
270 PRINT "wallpaper you'll need to":
    "decorate a room.":
280 PRINT :"You enter the room's":
    "dimensions and the number of"
290 PRINT "doors and windows.  The":
    "program calculates how much"
300 PRINT "paint/paper you'll need."
310 PRINT
320 GOSUB 6000
330 REM MAIN PROCESSING
340 REM GET ROOM SIZE, CALCULATE AREA
350 REM CALCULATE PAINT/PAPER REQUIREMENTS
360 REM AND THEN PRINT THE RESULTS
370 CALL CLEAR
380 GOSUB 1000
390 GOSUB 3000
400 GOSUB 4000
410 GOSUB 5000
420 INPUT "Process another room(Y/N)->":Y$
430 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 450
440 STOP
450 FOR I=1 TO 10
460 WALLS(I)=0
470 DOORS(I)=0
480 WINDOWS(I)=0
490 FOR J=1 TO 4
500 DOORSIZE(I,J)=0
510 NEXT J
520 FOR K=1 TO 6
530 WINSIZE(I,K)=0
540 NEXT K
550 NEXT I
560 CALL CLEAR
570 GOTO 330
1000 REM GET THE ROOM HEIGHT
1010 REM
1020 PRINT :"How high is the ceiling"
1030 INPUT " (0 to stop) -> ":CEILING
```

Listing 4-4—cont. Wallpaper/Paint Program

```
1040 IF CEILING<0 THEN 1000
1050 IF CEILING<>0 THEN 2000
1060 STOP
2000 REM NOW, GET THE WALL DIMENSIONS
2010 PRINT :"Enter all measurements":
     "in feet (round to the"
2020 PRINT "nearest foot)."
2030 INPUT "How many walls (0-10) ->":NUMWALLS
2040 IF NUMWALLS=0 THEN 1060
2050 IF NUMWALLS<0 THEN 2030
2060 IF NUMWALLS<=MAXWALL THEN 2090
2070 PRINT "You can only process data":" for";
     MAXWALL;"walls."
2080 GOTO 2030
2090 REM GET THE DIMENSIONS FOR EACH WALL
2100 REM AND THE NUMBER OF DOORS AND WINDOWS
2110 REM IF THE WINDOWS AREN'T LARGE, SKIP THEM
2120 REM
2130 PRINT :"Now you have to enter the":
     "number of doors and windows"
2140 PRINT "in each wall. You can have"
2150 PRINT "up to";MAXDOOR;"doors and";MAXWIN;"windows"
2160 PRINT "on each wall.": :"If the windows are not"
2170 PRINT "oversized, you don't have to":
     "include them.  You should"
2180 PRINT "include doors however.  If":
     "you have an archway to"
2190 PRINT "another room, estimate its":
     "size as a number of doors."
2200 PRINT
2210 GOSUB 6000
2220 REM
2230 REM NOW GET THE ROOM INFORMATION
2240 REM
2250 FOR I=1 TO NUMWALLS
2260 PRINT :"For wall ";I
2270 INPUT "How long is the wall ->":WALLS(I)
2280 IF WALLS(I)<=0 THEN 2270
2290 REM THE DOOR DATA
2300 REM
2310 INPUT "How many doors ->":DOORS(I)
2320 IF DOORS(I)<=0 THEN 2400
2330 IF DOORS(I)<=MAXDOOR THEN 2360
2340 PRINT "You can only process";MAXDOOR;"doors."
2350 GOTO 2310
2360 FOR J=1 TO DOORS(I)
2370 PRINT "For door ";J
2380 INPUT "How many square feet -> ":DOORSIZE(I,J)
2390 NEXT J
2400 REM FINALLY, THE WINDOW DATA
2410 REM
2420 INPUT "How many windows ->":WINDOWS(I)
2430 IF WINDOWS(I)<=0 THEN 2520
2440 IF WINDOWS(I)<=MAXWIN THEN 2470
2450 PRINT "You can only process";MAXWIN;"windows."
2460 GOTO 2400
2470 FOR J=1 TO WINDOWS(I)
```

Listing 4-4—cont. Wallpaper/Paint Program

```
2480 PRINT "For window ";J
2490 INPUT "How many square feet -> ":WINSIZE(I,J)
2500 NEXT J
2510 REM PROCESS THE NEXT WALL
2520 NEXT I
2530 RETURN
3000 REM CALCULATE THE TOTAL WALL AREA
3010 REM
3020 LENGTH=0
3030 DOORAREA=0
3040 WINAREA=0
3050 FOR I=1 TO NUMWALLS
3060 LENGTH=LENGTH+WALLS(I)
3070 FOR J=1 TO DOORS(I)
3080 DOORAREA=DOORAREA+DOORSIZE(I,J)
3090 NEXT J
3100 FOR K=1 TO WINDOWS(I)
3110 WINAREA=WINAREA+WINSIZE(I,K)
3120 NEXT K
3130 NEXT I
3140 REM REMOVE THE WINDOWS/DOORS
3150 REM
3160 ROOMAREA=CEILING*LENGTH
3170 AREA=RD(ROOMAREA-DOORAREA-WINAREA)
3180 DOORAREA=RD(DOORAREA)
3190 WINAREA=RD(WINAREA)
3200 PRINT :"Your room is";ROOMAREA;" sq ft"
3210 PRINT "with";DOORAREA;" sq ft of doors":"and";
3220 PRINT WINAREA;" sq ft of windows."
3230 PRINT
3240 RETURN
4000 REM CALCULATE HOW MUCH PAINT YOU NEED
4010 REM FOR COATS COVERING 100 THROUGH 400 SQ FT/GAL
4020 REM
4030 FOR I=1 TO 4
4040 PAINT(I)=(AREA+40)/(I*100)
4050 PAINT(I)=INT(PAINT(I)+.6)
4060 IF PAINT(I)>=1 THEN 4080
4070 PAINT(I)=1
4080 NEXT I
4090 REM NOW FOR THE WALLPAPER
4100 REM ONE ROLL COVERS 30 SQ FT
4110 REM ROLLS COME IN SINGLE/DOUBLE/TRIPLE LENGTHS
4120 REM
4130 FOR I=1 TO 3
4140 PAPER(I)=(AREA+15)/(I*30)
4150 PAPER(I)=INT(PAPER(I)+.5)
4160 IF PAPER(I)>=1 THEN 4180
4170 PAPER(I)=1
4180 NEXT I
4190 RETURN
5000 REM PRINT THE RESULTS
5010 REM
5020 GOSUB 6000
5030 CALL CLEAR
5040 PRINT "Your room is";AREA;" sq ft."
5050 PRINT : :"In paint, you'll need"
```

Listing 4-4—cont. Wallpaper/Paint Program

```
5060 FOR I=1 TO 4
5070 PRINT PAINT(I);" gals for ";STR$(I*100);
     " sq ft/gal"
5080 NEXT I
5090 PRINT : :"In paper, you'll need"
5100 FOR I=1 TO 3
5110 PRINT PAPER(I);" ";ROLL$(I);" rolls."
5120 NEXT I
5130 PRINT :"Don't forget to add gallons":
     "for multiple coats and extra"
5140 PRINT "rolls for matching large":"patterns."
5150 PRINT
5160 GOSUB 6000
5170 RETURN
6000 REM PAUSE
6010 PRINT "Press any key to continue.";
6020 CALL KEY(0,K,S)
6030 IF S=0 THEN 6020
6040 RETURN
```

YARN ESTIMATOR PROGRAM

What it does:

The Yarn Estimator program calculates how much yarn you need to cover a needlepoint canvas. There are many ways to calculate yarn coverage. We chose one which uses the length of yarn needed to cover 1 square inch of a particular canvas using a specific stitch.

You enter the canvas size and number of colors (up to 40) that you want to estimate. Then, choosing from two menus, you pick a canvas type/yarn type combination and stitch. The Yarn Estimator then tells you how much yarn you'll need to cover the entire canvas in a single color. If the yarn comes in ounces as well as yards (Persian does) you'll see how many ounces are needed, as well as yards.

If you have more than one color, the program asks you for a color name and percentage coverage for each color. You have a chance to change the coverages if you overestimate or underestimate the amounts of each color.

Since there are hundreds of needlepoint stitches and many types of canvas, we decided to give you a program that you can expand to include any stitches and canvas that you prefer. The table COVER shows the length of yarn needed. Each entry is based on a canvas/yarn and stitch combination.

The Yarn Estimator processes these canvas/yarn combinations:

- 5-mesh canvas/rug yarn
- 10-mesh canvas/light cotton yarn

- 10-mesh canvas/3-ply regular Persian yarn
- 12-mesh canvas/2-ply regular Persian yarn
- 14-mesh canvas/2-ply regular Persian yarn.

Since the amount of thread that covers a square inch depends on the stitch used, we included these stitches:

- 1/2 cross
- Continental
- Basketweave
- Bargello over 2 threads (uses 75% of 1/2 cross)
- Bargello over 4 threads (uses 50% of 1/2 cross)

What it shows you:

The Yarn Estimator program shows you how to look up information in a table. Since there are five possible stitches for each canvas/yarn combination, it was easy to make a table (COVER) that has canvas/yarn as one index (the first) and stitch as the other index. Each entry then tells you how many inches of yarn are needed to cover 1 square inch of the canvas using the stitch.

For example, if you have a 10-mesh canvas/3-ply Persian yarn and you want to use Continental stitch, you would get the COVER(3,2) entry which tells you that you need 50 inches of 3-ply Persian yarn to cover 1 square inch of 10-mesh canvas using the Continental stitch.

Since there are hundreds of stitches and many kinds of canvas, there has to be some way to tell what you're going to use. This program shows you how to use menus to simplify getting that information. It also makes it easy for you to add canvas types, stitches, and yarns that you use often.

What you can do:

The most common way to customize this program is to include your own favorite canvas, yarn, and stitch information. It's not at all difficult. Just follow these steps:

1. Find out how many inches of yarn you need to cover 1 square inch of your canvas.
2. Increase the array limits in line 280 and increase the values for MAXCAN and MAXST.
3. Add DATA statements after line 430 and before line 440. The first DATA statement should show the canvas type and the yarn (look at the other DATA statements in lines 340–430). The sec-

ond DATA statement shows the amount of yarn needed to cover 1 square inch in the stitches already used.

4. Fix the menus to include your new choices.
5. Include any new stitch names in the DATA statement at line 450.

If you include more stitches, make sure that you add another value to each of the DATA statements that show coverage (350, 370, etc.) so that your table is correct. If you don't know a value, add a 0.

For example, you want to add data for a Leaf stitch that you know uses 75 inches to cover 1 square inch of 10-mesh canvas using 3-ply Persian yarn. To do this you:

1. Increase MAXST to 6 (you added one stitch type).
2. Change the DIM statement to COVER(5,6) and STITCH$(6).
3. Add "Leaf stitch" to the end of DATA statement 450.
4. Fix the code in statements 1110–1180 to show the new stitch (Hint: make STOP be choice 7, Leaf be choice 6, and fix the two IF statements).
5. Run the program.

Table 4-9. Yarn Estimator Routines

Lines	Function
100-660	Initialize variables and screen.
670-750	Main processing.
1000-1270	Get the canvas type, number of colors, stitch type, and yarn type.
2000-2280	Calculate canvas area and length of yarn needed to cover the canvas.
3000-3310	Get the amount of each color, calculate the yarn needed for each color, and print the results.
4000-4040	Delay routine.

Table 4-10. Yarn Estimator Variables

Variable	Description
AREA	Area of canvas in square inches.
CANVAS$(5)	Descriptions of the canvas types.
CANYARN	Canvas type/yarn type indicator.
COLAMT(40,2)	The amount of each color needed.
COLAREA	The total area covered by the colors.
COLOR$(40)	The color names.
COVER(5,5)	The coverage table. Each entry shows the number of inches of yarn needed to cover one square inch of a canvas type with a specific stitch.
INCHES	The number of inches of yarn needed to cover the canvas.
LENGTH	Canvas length in inches.
MAXCAN	Maximum number of canvas types processed (currently 5).
MAXCOL	Maximum number of colors processed (currently 40).
MAXST	Maximum number of stitch types processed (currently 5).
NUMCOLOR	Number of colors to process for this canvas.
OUNCES2	Number of ounces of 2-ply Persian yarn (calculated as 1 ounce = 45 30-inch lengths and taking into account splitting each 3-ply into 2-ply strands).
OUNCES3	Number of ounces of 3-ply Persian yarn (calculated as 1 ounce = 45 30-inch lengths).
PLY2	Number of 30-inch lengths of 2-ply Persian needed to cover the canvas.
PLY3	Number of 30-inch lengths of 3-ply Persian needed to cover the canvas.
STITCH$(5)	Stitch descriptions.
TOTCOL	Total area covered by the individual colors.
WIDTH	Canvas width in inches.
YARDS	The number of yards of yarn needed to cover the canvas (calculated as INCHES/36).
YARN$(5)	Yarn type descriptions.

Listing 4-5. Yarn Estimator Program

```
100 REM THIS PROGRAM DETERMINES HOW MUCH YARN YOU
110 REM NEED TO COVER VARIOUS TYPES OF NEEDLEPOINT
120 REM CANVAS.
130 REM
140 REM STITCHES USED ARE:
150 REM 1/2 CROSS, CONTINENTAL, BASKETWEAVE,
    BARGELLO OVER 2
160 REM THREADS, BARGELLO OVER 4 THREADS.
170 REM
180 REM CANVAS SIZES ARE:
190 REM 5, 10, 12, 14 MESH.
200 REM
210 REM YARNS USED ARE:
220 REM RUG, 2-PLY PERSIAN, 3-PLY PERSIAN, COTTON.
230 REM
240 REM TABLE SHOWS HOW MANY INCHES OF YARN COVER
250 REM ONE SQUARE INCH OF CANVAS.
260 REM
270 OPTION BASE 1
280 DIM COVER(5,5),CANVAS$(5),STITCH$(5),YARN$(5)
290 DIM COLOR$(40),COLAMT(40,2)
300 MAXCAN=5
310 MAXST=5
320 MAXCOL=40
330 REM  DATA IS STORED FOR STITCHES IN ORDER ABOVE.
340 DATA "5-mesh","rug yarn"
350 DATA 20,30,33,10,15
360 DATA "10 mesh","light cotton"
370 DATA 43,67,74,22,33
380 DATA "10 mesh","3 ply regular persian"
390 DATA 35,50,55,18,27
400 DATA "12 mesh","2 ply regular persian"
410 DATA 10,60,64,20,30
420 DATA "14 mesh","2 ply regular persian"
430 DATA 45,70,77,22,34
440 DATA "1/2 cross","Continental","Basketweave"
450 DATA "2-Bargello","4-Bargello"
460 REM
470 REM FIRST READ IN THE YARN DATA
480 FOR I=1 TO MAXCAN
490 READ CANVAS$(I),YARN$(I)
500 FOR J=1 TO MAXST
510 READ COVER(I,J)
520 NEXT J
530 NEXT I
540 FOR I=1 TO MAXST
550 READ STITCH$(I)
560 NEXT I
570 CALL CLEAR
580 PRINT "NEEDLEPOINT CALCULATOR": : :
    "This program helps you to"
590 PRINT "calculate how much yarn":
    "you'll need to cover various"
600 PRINT "types of needlepoint canvas."
610 PRINT : :"You can use 5, 10, 12,":
    " or 14 mesh canvas"
620 PRINT "and rug, persian, or":"cotton yarns."
```

Listing 4-5—cont. Yarn Estimator Program

```
630 PRINT : :"Once you've found how much":
    "total yarn you need, you can"
640 PRINT "find out how much of each":
    "color is needed."
650 PRINT : :
660 GOSUB 4000
670 CALL CLEAR
680 GOSUB 1000
690 REM CALCULATE THE AREA AND THE AMOUNT OF THREAD
    NEEDED
700 GOSUB 2000
710 REM NOW, GET THE COLORS
720 GOSUB 3000
730 INPUT "Estimate another (Y/N) ->":Y$
740 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 670
750 STOP
1000 PRINT "Enter your canvas size":"  in inches."
1010 INPUT "Canvas width -> ":WIDTH
1020 IF WIDTH<=0 THEN 1010
1030 INPUT "Canvas length -> ":LENGTH
1040 IF LENGTH<=0 THEN 1030
1050 AREA=LENGTH*WIDTH
1060 INPUT "How many colors -> ":NUMCOLOR
1070 IF NUMCOLOR<=0 THEN 1060
1080 IF NUMCOLOR<MAXCOL THEN 1110
1090 PRINT "You can only process";MAXCOL;" colors."
1100 GOTO 1060
1110 PRINT :"Now, choose a stitch":" 1. 1/2 cross":
     " 2. continental"
1120 PRINT " 3. basketweave":" 4. 2-thread bargello"
1130 PRINT " 5. 4-thread bargello":" 6. STOP"
1140 PRINT
1150 INPUT "Your choice -> ":ST
1160 IF (ST<1)+(ST>6)THEN 1110
1170 IF ST<6 THEN 1190
1180 STOP
1190 PRINT "Choose a canvas/yarn":" 1. 5-mesh/rug yarn"
1200 PRINT " 2. 10-mesh/light cotton":
     " 3. 10-mesh/3-ply persian"
1210 PRINT " 4. 12-mesh/2-ply persian":
     " 5. 14-mesh/2-ply persian"
1220 PRINT " 6. STOP": :
1230 INPUT "Your choice -> ":CANYARN
1240 IF (CANYARN<1)+(CANYARN>6)THEN 1190
1250 IF CANYARN<6 THEN 1270
1260 STOP
1270 RETURN
2000 REM CALCULATE CANVAS AREA AND THREAD LENGTH
2010 AREA=LENGTH*WIDTH
2020 REM THE VALUE IN COVER() TELLS HOW MANY INCHES OF
2030 REM YARN ARE NEEDED TO COVER 1 SQ INCH OF CANVAS
2040 INCHES=AREA*COVER(CANYARN,ST)
2050 REM 3 PLY YARN COMES IN 30 IN LENGTHS
2060 REM 45 LENGTHS MAKE 1 OUNCE
2070 PLY3=INT(INCHES/30)+1
2080 OUNCES3=INT(PLY3/45)+1
2090 REM 3 PLY YARN SPLIT TO 2 PLY GIVES 1.5 STRANDS
```

Listing 4-5—cont. Yarn Estimator Program

```
2100 REM PER UNIT
2110 PLY2=INT(PLY3/1.5)+1
2120 OUNCES2=INT(PLY2/45)+1
2130 YARDS=INT(INCHES/36)+1
2140 REM NOW WRITE THE TOTALS YARN REQUIRED
2150 PRINT : :"For ";CANVAS$(CANYARN);" canvas"
2160 PRINT "using ";STITCH$(ST);" stitch":
     "you'll need ";COVER(CANYARN,ST);
2170 PRINT " inches of ";YARN$(CANYARN);
     " for each square inch."
2180 PRINT :"You have to cover";AREA;"sq in."
2190 PRINT "You need";YARDS;" yards."
2200 ON CANYARN GOTO 2260,2260,2210,2240,2240
2210 REM 10 MESH - PERSIAN
2220 PRINT "Or";PLY3;" strands.":"Or";
     OUNCES3;" ounces."
2230 GOTO 2260
2240 REM 12 MESH - PERSIAN
2250 PRINT "Or";PLY2;" strands.":"Or";
     OUNCES2;" ounces."
2260 PRINT
2270 GOSUB 4000
2280 RETURN
3000 REM GET THE COLORS
3010 IF NUMCOLOR>0 THEN 3030
3020 RETURN
3030 PRINT "Now, enter the colors":
     "and approximate percentage"
3040 PRINT "for each color.": :
     "If you need only one strand"
3050 PRINT "for a color, enter 0.":
     "Enter color amounts as"
3060 PRINT "percent of area (e.g. 10)": :
3070 TOTCOL=0
3080 PRINT "You have";NUMCOLOR;" colors": :
3090 FOR I=1 TO NUMCOLOR
3100 PRINT "Processing color";I
3110 INPUT "Enter the color name -> ":COLOR$(I)
3120 MSG$="How much "&COLOR$(I)&" -> "
3130 INPUT MSG$:COLAMT(I,1)
3140 COLAREA=(COLAMT(I,1)/100)*AREA
3150 TOTCOL=TOTCOL+COLAREA
3160 COLAMT(I,2)=INT(COLAMT(I,1)*YARDS)/100
3170 NEXT I
3180 TOTCOL=INT(TOTCOL*100)/100
3190 PRINT "You entered enough colors":"to cover";
     TOTCOL;"sq in."
3200 INPUT
     "Do you want to re-enter the colors (Y/N) -> ":Y$
3210 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 3070
3220 REM PRINT AMOUNT FOR EACH COLOR
3230 LINES=0
3240 FOR I=1 TO NUMCOLOR
3250 PRINT COLOR$(I);COLAMT(I,2);"yds"
3260 LINES=LINES+1
3270 IF LINES<20 THEN 3300
3280 GOSUB 4000
```

Listing 4-5—cont. Yarn Estimator Program

```
3290 LINES=1
3300 NEXT I
3310 RETURN
4000 REM PAUSE
4010 PRINT "Press any key to continue."
4020 CALL KEY(0,K,S)
4030 IF S=0 THEN 4020
4040 RETURN
```

5

Personal Records

This chapter gives you these programs to help you keep track of your personal records:

- ADDRESS BOOK: Store your address book information, search for names, add data.
- CARDLIST: Keep track of who sent you cards and whom you sent cards to.
- TRIP PLANNER: Find out how much it's going to cost you for food, fuel, lodging, and entertainment.

ADDRESS BOOK PROGRAM

What it does:

The Address Book program manages a set of names, addresses, and phone numbers for you. Through a series of menus, you can:

1. Print the data in your address book.
2. Remove a name from your book.
3. Add a name to your book.
4. Change a name in your book.
5. Save your data to a tape/disk/other.
6. Read data from a tape/disk/other.
7. Search your book for a name.

You enter the following data for each entry. The information is stored as a character string containing all the data for an entry:

- first name
- last name

- street address
- city
- state
- zip code
- telephone number

What it shows you:

The Address Book program shows you one way to use and store string data. The data for a single entry (name, address, etc.) could be stored either as a single 89-character string or as 7 shorter strings.

Since each string array element in an array has a 3-byte overhead (2 bytes to point to the string data and 1 byte for the string length), we decided to "pack" the data into one string so that we would be able to store more entries in memory.

The D$ data array has 75 elements of 89 bytes (6675 bytes of actual data). With the 3-byte overhead, each element occupies 92 bytes, or 6900 bytes total. If the strings were stored instead as 7 arrays of shorter strings, the total for the 75 elements would be 8250 bytes (75 × 3 bytes per string × 7 strings + 75 × 89 bytes of data in the 7 shorter strings).

This program also shows you how you can use the same code for different data collection. Data for the names/addresses has to be collected for a new entry or for a changed entry, so this program uses an indicator (REQ) and a single routine to get data for new/changed entries. When the statements at 7000 are executed, the value of REQ determines which data (name, address, phone, etc.) to request.

What you can do:

If you have more memory and the Extended BASIC cartridge, you can increase the number of entries you can store in your address list. Remember to change the D$ array limit and the value for MAXADD (this should be the same as the D$ array limit).

If you have a printer, you might want to format a nice listing of your data. An easy way to do this is to add another choice to the main menu and GOSUB to a routine that prints the data.

You may find that the names and addresses that you store have to be longer than the values that we've chosen. If you lengthen the information, be sure that you adjust all the string sizes and beginning positions in the SEG$ functions that define each part of the data. Be sure you also change the statements that adjust the length of the data.

The data is currently searched on a last name basis. You can

change the search to look for an address, first name, or some other choice. Or, you can change the program to let *you* decide what to search for, like first name or zip code. (Hint: add a menu in the search routine at statements 6000–6100 to decide which item to look for.)

Table 5-1. Address Book Routines

Lines	Function
100-300	Initialize variables and screen.
310-410	Main menu processing.
1000-1170	Print address data to the screen.
2000-2220	Remove an entry from the address data.
3000-3230	Add data to the address book.
4000-4870	Change data in the address book.
5000-5760	Load/Save address book data.
6000-6100	Search for an entry in the address book.
7000-7340	Get the name/address/phone number data for new entries.
8000-8060	Print an entry to the terminal.
8200-8370	Sort the data in last name order.
8500-8730	Find a record in the data.
9000-9040	Delay routine.

Table 5-2. Address Book Variables

Variable	Description
ADD	Number of addresses in the current address book.
B$	A string of blanks used to pad data items to appropriate lengths.
D$(75)	The address book data.
LINES	Number of lines printed.
LST$	Last name used as a search argument.
MAXADD	Maximum number of addresses you can store in your book.
NUMREC	Number of records written to a tape/disk/other.
REQ	Indicator used to determine which piece of data to read (1=first name, 2=last name, etc.).
TMP$	A temporary string variable used to format the D$ data string.
T$	A temporary string variable used to format the D$ data string.

Listing 5-1. Address Book Program

```
100 REM  ADDRESS BOOK PROGRAM
110 REM
120 REM STORES FIRST NAME/LAST NAME/STREET/CITY/
    STATE/ZIP/PHONE
130 REM IN 89-CHARACTER STRINGS
140 REM LENGTHS ARE    15/15/25/15/2/5/12
150 REM DATA BEGINS AT 1/16/31/56/71/73/78
160 REM
170 OPTION BASE 1
180 DIM D$(75)
190 B$="                              "
200 MAXADD=75
210 ADD=0
220 CALL CLEAR
230 PRINT "** Address Book Program **"
240 PRINT : : : :"This program helps you to"
250 PRINT "manage your address book.": :
    "You can save your data on"
260 PRINT "a cassette or disk.": :
    "You can add/delete names"
270 PRINT "or change addresses or":"phone numbers."
280 PRINT : :
290 GOSUB 9000
300 CALL CLEAR
310 PRINT :"Address Book Options:": :
    " 1 Print contents"
320 PRINT " 2 Remove name from book":
    " 3 Add name to book"
330 PRINT " 4 Change name":" 5 Save data":
    " 6 Load data"
340 PRINT " 7 Search for a name":" 8 STOP"
350 PRINT : :
360 INPUT "YOUR CHOICE -> ":ANS
370 IF (ANS<1)+(ANS>8)THEN 310
380 IF ANS<8 THEN 400
390 STOP
400 ON ANS GOSUB 1000,2000,3000,4000,5000,5160,6000
410 GOTO 310
1000 REM FIRST, SEE IF THERE'S ANY DATA
1010 IF ADD>0 THEN 1040
1020 PRINT :"You don't have any data yet."
1030 RETURN
1040 REM PRINT THE BOOK
1050 LINES=0
1060 CALL CLEAR
1070 FOR I=1 TO ADD
1080 FND=I
1090 GOSUB 8000
1100 LINES=LINES+10
1110 IF LINES<20 THEN 1140
1120 GOSUB 9000
1130 LINES=0
1140 NEXT I
1150 IF LINES=0 THEN 1170
1160 GOSUB 9000
1170 RETURN
2000 REM FIRST, SEE IF THERE'S ANY DATA
```

Listing 5-1—cont. Address Book Program

```
2010 IF ADD>0 THEN 2040
2020 PRINT :"You don't have any data yet."
2030 RETURN
2040 PRINT :"Removing entry from book"
2050 GOSUB 8500
2060 IF FND<>0 THEN 2110
2070 INPUT "Do you want to re-enter the name? (Y/N)->"
     :Y$
2080 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 2050
2090 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 2220
2100 GOTO 2070
2110 GOSUB 8000
2120 INPUT "Remove this record (Y/N)->":Y$
2130 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 2160
2140 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 2220
2150 GOTO 2120
2160 REM REMOVE RECORD
2170 FOR I=FND TO ADD-1
2180 D$(I)=D$(I+1)
2190 NEXT I
2200 PRINT "Record ";FND;" removed"
2210 ADD=ADD-1
2220 RETURN
3000 REM ADD NAME TO DATA
3010 ADD=ADD+1
3020 D$(ADD)=""
3030 FOR I=1 TO 7
3040 REQ=I
3050 GOSUB 7000
3060 D$(ADD)=D$(ADD)&TMP$
3070 NEXT I
3080 FND=ADD
3090 GOSUB 8000
3100 PRINT
3110 INPUT "Add this record (Y/N) -> ":Y$
3120 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 3150
3130 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 3180
3140 GOTO 3100
3150 IF ADD<=MAXADD THEN 3190
3160 PRINT :"Too many entries.":"Try making two books."
3170 RETURN
3180 ADD=ADD-1
3190 PRINT
3200 INPUT "Add another name (Y/N)->":Y$
3210 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 3000
3220 GOSUB 8200
3230 RETURN
4000 REM CHANGE A NAME
4010 REM FIRST, SEE IF THERE'S ANY DATA
4020 IF ADD>0 THEN 4050
4030 PRINT :"You don't have any data yet."
4040 RETURN
4050 REM FIRST, FIND WHICH NAME
4060 GOSUB 8500
4070 IF FND>0 THEN 4100
4080 INPUT "Re-enter the name (Y/N)->":Y$
4090 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4010
```

Listing 5-1—cont. Address Book Program

```
     ELSE 4870
4100 PRINT :"You can change:":" 1 first name":
     " 2 last name"
4110 PRINT " 3 street address":" 4 city":" 5 state"
4120 PRINT " 6 zip code":" 7 phone number":" 8 DONE": :
4130 INPUT "Your choice -> ":ANS
4140 IF (ANS<1)+(ANS>8)THEN 4100
4150 IF ANS=8 THEN 4870
4160 ON ANS GOTO 4170,4270,4370,4470,4570,4670,4770
4170 REM CHANGE FIRST NAME
4180 PRINT :"First name is: ";SEG$(D$(FND),1,15)
4190 INPUT "Change it (Y/N) ->":Y$
4200 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4220
4210 GOTO 4100
4220 REQ=1
4230 GOSUB 7000
4240 T$=SEG$(D$(FND),1,15)&TMP$
4250 D$(FND)=T$
4260 GOTO 4100
4270 REM CHANGE LAST NAME
4280 PRINT :"Last name is: ";SEG$(D$(FND),1,15):
4290 INPUT "Change it (Y/N) ->":Y$
4300 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4320
4310 GOTO 4100
4320 REQ=2
4330 GOSUB 7000
4340 T$=SEG$(D$(FND),1,15)&TMP$&SEG$(D$(FND),31,255)
4350 D$(FND)=T$
4360 GOTO 4100
4370 REM CHANGE STREET ADDRESS
4380 PRINT :"Address is: ";SEG$(D$(FND),31,25)
4390 INPUT "Change it (Y/N) ->":Y$
4400 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4420
4410 GOTO 4100
4420 REQ=3
4430 GOSUB 7000
4440 T$=SEG$(D$(FND),1,30)&TMP$&SEG$(D$(FND),56,255)
4450 D$(FND)=T$
4460 GOTO 4100
4470 REM CHANGE CITY
4480 PRINT :"City is: ";SEG$(D$(FND),56,15)
4490 INPUT "Change it (Y/N) ->":Y$
4500 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4520
4510 GOTO 4100
4520 REQ=4
4530 GOSUB 7000
4540 T$=SEG$(D$(FND),1,55)&TMP$&SEG$(D$(FND),71,255)
4550 D$(FND)=T$
4560 GOTO 4100
4570 REM CHANGE STATE
4580 PRINT :"City is: ";SEG$(D$(FND),71,2)
4590 INPUT "Change it (Y/N) ->":Y$
4600 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4620
4610 GOTO 4100
4620 REQ=5
4630 GOSUB 7000
4640 T$=SEG$(D$(FND),1,70)&TMP$&SEG$(D$(FND),73,255)
```

Listing 5-1—cont. Address Book Program

```
4650 D$(FND)=T$
4660 GOTO 4100
4670 REM CHANGE ZIP
4680 PRINT :"Zip code is: ";SEG$(D$(FND),73,5)
4690 INPUT "Change it (Y/N) ->":Y$
4700 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4720
4710 GOTO 4100
4720 REQ=6
4730 GOSUB 7000
4740 T$=SEG$(D$(FND),1,72)&TMP$&SEG$(D$(FND),78,255)
4750 D$(FND)=T$
4760 GOTO 4100
4770 REM CHANGE PHONE NUMBER
4780 PRINT :"Phone number is: ";SEG$(D$(FND),78,12)
4790 INPUT "Change it (Y/N) ->":Y$
4800 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 4820
4810 GOTO 4100
4820 REQ=7
4830 GOSUB 7000
4840 T$=SEG$(D$(FND),1,77)&TMP$
4850 D$(FND)=T$
4860 GOTO 4100
4870 RETURN
5000 REM SAVE THE DATA
5010 REM FIRST, SEE IF THERE'S ANY DATA
5020 IF ADD>0 THEN 5050
5030 PRINT :"You don't have any data yet."
5040 RETURN
5050 RW=1
5060 FNUM=2
5070 GOSUB 5500
5080 GOSUB 8200
5090 PRINT #2:ADD
5100 FOR I=1 TO ADD
5110 PRINT #2:D$(I)
5120 NEXT I
5130 PRINT :ADD;" entries written."
5140 CLOSE #2
5150 RETURN
5160 REM LOAD THE DATA
5170 RW=2
5180 FNUM=1
5190 GOSUB 5500
5200 IF WDEV=0 THEN 5340
5210 IF EOF(1)THEN 5270
5220 INPUT #1:NUMREC
5230 FOR I=1 TO NUMREC
5240 INPUT #1:D$(I)
5250 ADD=ADD+1
5260 NEXT I
5270 PRINT :ADD;" entries read."
5280 CLOSE #1
5290 IF ADD=NUMREC THEN 5340
5300 PRINT "Check your data."
5310 PRINT :"Your tape/disk says it":"has";NUMREC;
     " records."
5320 PRINT :ADD;" records were":" actually read."
```

Listing 5-1—cont. Address Book Program

```
5330 PRINT
5340 RETURN
5500 PRINT "Address book can be on ":" 1 cassette":
     " 2 disk":" 3 other"
5510 PRINT " 4 none"
5520 INPUT "Where is your data -> ":ANS
5530 IF (ANS<1)+(ANS>4)THEN 5500
5540 WDEV=0
5550 IF ANS=4 THEN 5610
5560 WDEV=ANS
5570 ON ANS GOTO 5580,5630,5710
5580 ON RW GOTO 5590,5610
5590 OPEN #FNUM:"CS1",OUTPUT,FIXED,VARIABLE 128
5600 RETURN
5610 OPEN #FNUM:"CS1",INPUT ,FIXED,VARIABLE 128
5620 RETURN
5630 PRINT "Enter the file name":"as DSKn.filename"
5640 INPUT "Your filename -> ":FILE$
5650 ON RW GOTO 5660,5680
5660 OPEN #FNUM:FILE$,OUTPUT,INTERNAL,VARIABLE 128
5670 RETURN
5680 OPEN #FNUM:FILE$,INPUT ,INTERNAL,VARIABLE 128
5690 RETURN
5700 PRINT "Enter the file name":"as unit.filename"
5710 INPUT "Your filename ->":FILE$
5720 ON RW GOTO 5730,5750
5730 OPEN #FNUM:FILE$,OUTPUT,INTERNAL,VARIABLE 128
5740 RETURN
5750 OPEN #FNUM:FILE$,INPUT ,INTERNAL,VARIABLE 128
5760 RETURN
6000 REM SEARCH FOR NAME
6010 IF ADD>0 THEN 6040
6020 PRINT :"You don't have any data yet."
6030 RETURN
6040 GOSUB 8500
6050 IF FND<>0 THEN 6090
6060 REM RECORD NOT FOUND
6070 INPUT "Do you want to re-enter the name? (Y/N)->"
     :Y$
6080 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 6000
     ELSE 6100
6090 GOSUB 8000
6100 RETURN
7000 REM GET DATA
7010 REM MAKE SURE EACH PIECE IS THE CORRECT LENGTH
7020 ON REQ GOTO 7030,7090,7110,7170,7190,7240,7290
7030 INPUT "First name-> ":TMP$
7040 IF LEN(TMP$)>15 THEN 7070
7050 TMP$=TMP$&SEG$(B$,1,15-LEN(TMP$))
7060 RETURN
7070 TMP$=SEG$(TMP$,1,15)
7080 RETURN
7090 INPUT "Last name-> ":TMP$
7100 IF LEN(TMP$)>15 THEN 7070 ELSE 7050
7110 INPUT "Street address-> ":TMP$
7120 IF LEN(TMP$)>25 THEN 7150
7130 TMP$=TMP$&SEG$(B$,1,25-LEN(TMP$))
```

Listing 5-1—cont. Address Book Program

```
7140 RETURN
7150 TMP$=SEG$(TMP$,1,2)
7160 RETURN
7170 INPUT "City-> ":TMP$
7180 IF LEN(TMP$)>15 THEN 7070 ELSE 7050
7190 REM GET STATE
7200 INPUT "State-> ":TMP$
7210 IF LEN(TMP$)=2 THEN 7340
7220 PRINT "Use a 2-character state."
7230 GOTO 7200
7240 REM GET ZIP CODE
7250 INPUT "Zip code-> ":TMP$
7260 IF LEN(TMP$)=5 THEN 7340
7270 PRINT "Zip codes must be 5 digits."
7280 GOTO 7250
7290 REM GET PHONE NUMBER
7300 INPUT "Phone number-> ":TMP$
7310 IF LEN(TMP$)>12 THEN 7300
7320 IF LEN(TMP$)=12 THEN 7340
7330 TMP$=TMP$&SEG$(B$,1,12-LEN(TMP$))
7340 RETURN
8000 REM PRINT RECORD
8010 PRINT :"FIRST NAME: ";SEG$(D$(FND),1,15):
     "LAST NAME: ";SEG$(D$(FND),16,15)
8020 PRINT "STREET: ";SEG$(D$(FND),31,25):"CITY: ";
     SEG$(D$(FND),56,15)
8030 PRINT "STATE: ";SEG$(D$(FND),71,2):"ZIP: ";
     SEG$(D$(FND),73,5)
8040 PRINT "PHONE: ";SEG$(D$(FND),78,10)
8050 PRINT
8060 RETURN
8200 REM SORT DATA
8210 IF ADD>0 THEN 8230
8220 RETURN
8230 PRINT "Sorting data"
8240 TOP=ADD-1
8250 FLAG=0
8260 FOR I=1 TO TOP
8270 IF SEG$(D$(I),16,15)<=SEG$(D$(I+1),16,15)THEN 8320
8280 FLAG=1
8290 T$=D$(I)
8300 D$(I)=D$(I+1)
8310 D$(I+1)=T$
8320 NEXT I
8330 REM SEE IF ANY ELEMENTS WERE CHANGED
8340 IF FLAG=0 THEN 8370
8350 TOP=TOP-1
8360 GOTO 8250
8370 RETURN
8500 REM FIND RECORD
8510 REM FIRST, SEE IF THERE'S ANY DATA
8520 REM
8530 IF ADD>0 THEN 8570
8540 PRINT :"There are no names in your":
     " address book yet."
8550 FND=0
8560 RETURN
```

Listing 5-1—cont. Address Book Program

```
8570 REM GET THE LAST NAME FOR THE RECORD
8580 REM
8590 INPUT "Enter last name -> ":LST$
8600 IF LEN(LST$)<15 THEN 8630
8610 LST$=SEG$(LST$,1,15)
8620 GOTO 8640
8630 LST$=LST$&SEG$(B$,1,15-LEN(LST$))
8640 REM FIND PLACE IN ARRAY
8650 BEG=1
8660 FND=0
8670 FOR I=BEG TO ADD
8680 IF SEG$(D$(I),16,15)<>LST$ THEN 8710
8690 FND=I
8700 RETURN
8710 NEXT I
8720 PRINT "No entry for ";LST$
8730 RETURN
9000 REM PAUSE
9010 PRINT : :"Press any key to continue."
9020 CALL KEY(0,K,S)
9030 IF S=0 THEN 9020
9040 RETURN
```

CARDLIST PROGRAM

What it does:

The CardList program is a menu-driven program that helps you keep track of who sent you holiday cards and to whom you sent cards.

By simply choosing from its main menu, the program lets you do several different types of processing on your card data, such as:

1. Create a new card list
2. Load an existing card list from tape, disk, or other medium
3. Save the list currently in the computer to tape, disk, or other medium
4. Search the current list for a name (last name, first name, both first and last names)
5. Add names to your list
6. Remove names from your list
7. Print your list.

When you're entering names in your card list, the program first asks you for the person's last name and first name. Then you mark the name as:

1. You sent the person a card
2. You received a card from the person
3. You sent a card to and received a card from the person
4. Don't include this name in the list.

Since the program was written to fit into the 16K memory and uses the TI console BASIC, you can have a maximum of 150 names in your list. If your list is longer than 150, make several smaller lists.

What it shows you:

This CardList program shows you how to use menus to determine what processing to perform.

You'll notice that there's only a slight difference in the processing for creating a new list and for adding names to an existing list. The only real difference is the number of names already in the list.

You'll also see that most of the menus let you get out of the processing if you change your mind or if you get somewhere by mistake. For example, if you enter a name and then decide that you don't want to include that person, you can skip the entry by choosing "skip entry" from the menu.

The code for finding out where to read from/write to prompts you to enter the filename in the format appropriate to the device you choose. You use a different type of filename for a cassette and a disk. Since you can only write to cassette 2 (CS2), the CardList program doesn't let you read from it. You can both read from and write to cassette 1 (CS1).

The section that searches for a name to be removed from the list (lines 5000–5230) lets you continue to search your list after the first name is found. You might have several people with the same last name and want to remove only one of them from your list.

The list information is sorted into alphabetical order by last name. This makes it easier to find a name in your list when you print it out and makes it faster to find a name when you're searching the list. Once you've found a name past the "search name" you've chosen, you know that your "search name" can't be in the list.

What you can do:

Since the program stores only 150 names, you can expand the number of names if you have more memory and the Extended BASIC cartridge. (You can't address more than the 16K console memory with the console BASIC.) Remember to increase the SENT and NAME$ array sizes and set MAXCARDS to your new limit.

You may have a different storage device (like a Hexbus Wafertape). If so, you can add code to read from/write to the new device. Remember to add selections for the new device in the menus in sections 2000 and 3000 and to add appropriate OPEN statements in these sections.

You might want to write to a printer as well as to the screen. The Mortgage program in this book shows you an easy way to write to both a printer and the screen.

Table 5-3. CardList Routines

Lines	Function
100-230	Initialize variables and screen.
240-340	Main menu processing.
1000-1280	Create a new list or add names to an existing list.
2000-2250	Read a list from tape/disk/other.
3000-3270	Write a list to tape/disk/other.
4000-4360	Search the current list for a name.
5000-5230	Remove a name from the list. Continue searching (use 4000–4360) after finding the first occurrence of the name.
6000-6430	Print the list at the screen.
7000-7220	Sort the list into last name order.
8000-8030	Delay routine.

Table 5-4. CardList Variables

Variable	Description
BEGIN	Where to start searching for a name in the list.
CARDS	Number of names in the current list in memory.
FOUND	Found/not found flag.
LAST	Used to indicate whether or not to search the list again after finding the first occurrence of a name.
LINES	Number of lines printed at screen.
MAXCARDS	Maximum number of entries allowed in the list.
NAME$(150)	Names in the list.
SEARCH$	Name to search for in the list.
SENT(150)	Sent/Received/Both indicator for each name in the list.

Listing 5-2. CardList Program

```
100 REM THIS PROGRAM HELPS YOU TO KEEP TRACK OF YOUR
110 REM CHRISTMAS CARD LIST
120 REM
130 OPTION BASE 1
140 DIM SENT(150),NAME$(150)
150 CARDS=0
160 MAXCARDS=150
170 CALL CLEAR
180 PRINT "This program helps you to":
    "manage your Christmas card":"list."
190 PRINT :"You can easily keep track":
    "of cards that you sent and"
200 PRINT "cards that you received."
210 PRINT : :
220 GOSUB 8000
230 CALL CLEAR
240 PRINT :"You can:":" 1  Create a new list":
```

Listing 5-2—cont. CardList Program

```
    " 2  Load an existing list"
250 PRINT " 3  Save your list":" 4  Search for a name":
    " 5  Add names"
260 PRINT " 6  Remove names":" 7  Print your list":
    " 8  STOP"
270 PRINT
280 INPUT "Your choice -> ":ANS
290 IF (ANS<1)+(ANS>8)THEN 240
300 IF ANS<8 THEN 330
310 STOP
320 REMFLAG=0
330 ON ANS GOSUB 1000,2000,3000,4000,1030,5000,6000
340 GOTO 240
1000 REM CREATE A NEW LIST
1010 REM
1020 CARDS=0
1030 CARDS=CARDS+1
1040 IF CARDS<=MAXCARDS THEN 1090
1050 CARDS=CARDS-1
1060 PRINT :"You can't enter more than ":MAXCARDS;
     " names."
1070 PRINT "Try making two lists."
1080 RETURN
1090 PRINT
1100 PRINT "Enter the name (DONE":"  when finished)"
1110 INPUT "Last name -> ":ANS$
1120 IF SEG$(ANS$,1,4)="DONE" THEN 1260
1130 NAME$(CARDS)=ANS$
1140 INPUT "First name(s) -> ":ANS$
1150 NAME$(CARDS)=NAME$(CARDS)&", "&ANS$
1160 PRINT
1170 PRINT :"Mark this as:":" 1 card sent":
     " 2 card received":" 3 both"
1180 PRINT " 4 skip entry": :
1190 INPUT "Your choice -> ":ANS
1200 IF (ANS<1)+(ANS>4)THEN 1170
1210 IF ANS=4 THEN 1240
1220 SENT(CARDS)=ANS
1230 GOTO 1030
1240 CARDS=CARDS-1
1250 GOTO 1030
1260 CARDS=CARDS-1
1270 GOSUB 7000
1280 RETURN
2000 REM LOAD AN EXISTING LIST
2010 REM
2020 PRINT :"Where is your data:":" 1  cassette":
     " 2  disk":" 3  other"
2030 PRINT " 4  don't load data"
2040 PRINT
2050 INPUT "Your choice -> ":ANS
2060 IF (ANS<1)+(ANS>4)THEN 2000
2070 IF ANS=4 THEN 2250
2080 ON ANS GOTO 2090,2110,2150
2090 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
2100 GOTO 2180
2110 PRINT "Enter the file name":"as DSKn.filename"
```

Listing 5-2—cont. CardList Program

```
2120 INPUT "Your file -> ":FILENAME$
2130 OPEN #1:FILENAME$,INPUT ,INTERNAL
2140 GOTO 2180
2150 PRINT :"Enter the file name":"as unit.filename"
2160 INPUT "Your file -> ":FILENAME$
2170 OPEN #1:FILENAME$,INPUT ,INTERNAL
2180 REM NOW READ THE LIST
2190 INPUT #1:CARDS
2200 FOR I=1 TO CARDS
2210 INPUT #1:NAME$(I),SENT(I)
2220 NEXT I
2230 PRINT :CARDS;" records read."
2240 CLOSE #1
2250 RETURN
3000 REM SAVE THE LIST
3010 REM
3020 PRINT "Where do you want to":"save your data":
     " 1  cassette"
3030 PRINT " 2  disk":" 3  other":" 4  don't save data"
3040 INPUT "Your choice -> ":ANS
3050 IF (ANS<1)+(ANS>4)THEN 3000
3060 IF ANS=4 THEN 3270
3070 ON ANS GOTO 3080,3130,3170
3080 PRINT
3090 INPUT "Cassette 1 or 2 -> ":CAS
3100 IF (CAS<1)+(CAS>2)THEN 3080
3110 OPEN #2:"CS"&STR$(CAS),OUTPUT,INTERNAL,FIXED 128
3120 GOTO 3200
3130 PRINT :"Enter the file name":"as DSKn.filename"
3140 INPUT "Your file -> ":FILENAME$
3150 OPEN #2:FILENAME$,OUTPUT,INTERNAL
3160 GOTO 3200
3170 PRINT :"Enter the file name":"as unit.filename"
3180 INPUT "Your file -> ":FILENAME$
3190 OPEN #2:FILENAME$,OUTPUT,INTERNAL
3200 REM NOW WRITE THE LIST
3210 PRINT #2:CARDS
3220 FOR I=1 TO CARDS
3230 PRINT #2:NAME$(I),SENT(I)
3240 NEXT I
3250 CLOSE #2
3260 PRINT :CARDS;" records written."
3270 RETURN
4000 REM SEARCH FOR A NAME
4010 IF CARDS>0 THEN 4050
4020 PRINT :"There is no data in your":" list yet."
4030 RETURN
4040 REM
4050 PRINT :"Depending on how you enter":
     "the name, you can search"
4060 PRINT " for a match on:":" last name only"
4070 PRINT " first name only":
     " ""last name, first name"""
4080 PRINT " DONE to end"
4090 PRINT
4100 INPUT "Enter the name -> ":SEARCH$
4110 IF SEARCH$="DONE" THEN 4240
```

Listing 5-2—cont. CardList Program

```
4120 BEGIN=1
4130 FOUND=0
4140 ANY=0
4150 FOR I=BEGIN TO CARDS
4160 LAST=I
4170 IF POS(NAME$(I),SEARCH$,1)=0 THEN 4210
4180 FOUND=I
4190 ANY=1
4200 GOTO 4250
4210 NEXT I
4220 IF ANY>0 THEN 4240
4230 PRINT :"Name """;SEARCH$;""" not found."
4240 RETURN
4250 IF SENT(FOUND)=2 THEN 4310
4260 PRINT :"You sent a card to"
4270 IF SENT(FOUND)=1 THEN 4320
4280 IF SENT(FOUND)=2 THEN 4310
4290 PRINT "And you received a card from"
4300 GOTO 4320
4310 PRINT "You received a card from"
4320 PRINT NAME$(FOUND)
4330 BEGIN=FOUND+1
4340 IF REMFLAG=1 THEN 4360
4350 GOTO 4150
4360 RETURN
5000 REM REMOVE A NAME
5010 REMFLAG=1
5020 GOSUB 4000
5030 IF FOUND=0 THEN 5150
5040 PRINT :"Remove "&NAME$(FOUND)
5050 INPUT " (Y/N)-> ":Y$
5060 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 5090
5070 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 5160
5080 GOTO 5040
5090 FOR I=FOUND TO CARDS-1
5100 NAME$(I)=NAME$(I+1)
5110 SENT(I)=SENT(I+1)
5120 NEXT I
5130 CARDS=CARDS-1
5140 PRINT :SEARCH$;" removed."
5150 RETURN
5160 REM SEE IF SEARCH FOR ANOTHER HIT
5170 IF LAST>=CARDS THEN 5230
5180 INPUT "Continue searching (Y/N) -> ":Y$
5190 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")THEN 5200
5200 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")THEN 5230
5210 GOSUB 4130
5220 GOTO 5030
5230 RETURN
6000 REM PRINT THE LIST
6010 REM
6020 IF CARDS>0 THEN 6050
6030 PRINT :"There is no data in your":" list yet."
6040 RETURN
6050 REM PRINT HEADINGS
6060 CALL CLEAR
6070 GOSUB 6400
```

Listing 5-2—cont. CardList Program

```
6080 REM NOW, PRINT THE DATA
6090 FOR I=1 TO CARDS
6100 IF LINES<20 THEN 6130
6110 GOSUB 8000
6120 GOSUB 6400
6130 SFLAG$=" "
6140 RFLAG$=" "
6150 IF SENT(I)=2 THEN 6180
6160 SFLAG$="X"
6170 IF SENT(I)=1 THEN 6200
6180 RFLAG$="X"
6190 IF LEN(NAME$(I))>16 THEN 6230
6200 PRINT TAB(3);SFLAG$;TAB(8);RFLAG$;TAB(12);NAME$(I)
6210 LINES=LINES+1
6220 GOTO 6370
6230 N=POS(NAME$(I),",",1)
6240 IF N>16 THEN 6330
6250 TMPL$=SEG$(NAME$(I),1,N)
6260 TMPF$=SEG$(NAME$(I),N+2,255)
6270 IF LEN(TMPF$)>16 THEN 6310
6280 PRINT TAB(3);SFLAG$;TAB(8);RFLAG$;TAB(12);TMPL$;
     TAB(12);TMPF$
6290 LINES=LINES+2
6300 GOTO 6370
6310 TABST=12
6320 IF LEN(TMPF$)<28 THEN 6340
6330 TMPF$=SEG$(TMPF$,1,28)
6340 TABST=1
6350 PRINT TAB(3);SFLAG$;TAB(8);RFLAG$;TAB(12);TMPL$;
     TAB(TABST);TMPF$
6360 LINES=LINES+2
6370 NEXT I
6380 GOSUB 8000
6390 RETURN
6400 REM PRINT THE TITLES
6410 LINES=2
6420 PRINT "SENT ";"RECD ";TAB(15);"NAME":
     "==== ==== ================="
6430 RETURN
7000 REM SORT THE LIST IN NAME ORDER
7010 REM
7020 IF CARDS>0 THEN 7040
7030 RETURN
7040 PRINT :"Sorting data."
7050 CHG=0
7060 LAST=CARDS-1
7070 FOR I=1 TO LAST
7080 IF NAME$(I+1)>=NAME$(I)THEN 7160
7090 CHG=1
7100 TMP$=NAME$(I)
7110 NAME$(I)=NAME$(I+1)
7120 NAME$(I+1)=TMP$
7130 HOLD=SENT(I)
7140 SENT(I)=SENT(I+1)
7150 SENT(I+1)=HOLD
7160 NEXT I
7170 REM IF NOTHING CHANGED, RETURN
```

Listing 5-2—cont. CardList Program

```
7180 IF CHG=0 THEN 7220
7190 CHG=0
7200 LAST=LAST-1
7210 GOTO 7070
7220 RETURN
8000 PRINT "Press any key to continue."
8010 CALL KEY(0,K,S)
8020 IF S=0 THEN 8010
8030 RETURN
```

TRIP PLANNER PROGRAM

What it does:

This is a *menu-driven* program where you select what you want to do from a numbered list printed on the screen. It's very easy to use a lasting up to 25 days.

The program keeps track of each day's:

- destination
- distance
- cost of lodging
- cost of food
- cost of tolls
- cost of gasoline
- other expenses

You can get reports by day and a total for the entire trip. You can save the trip data to tape or disk, recall it later, and add, delete, or change items in it.

What it shows you:

This is a *menu-driven* program where you select what you want to do from a numbered list printed in the screen. It's very easy to use a menu-driven program and the technique fits well with a modular program design.

We also do some file input/output in this program. If you're interested in reading and writing to tape or disk from BASIC, you can examine the appropriate routines in this Trip Planner to see how it's done.

What you can do:

One of the easiest changes to make is to increase the maximum trip length from the arbitrary 25 we chose. If you want to do this, pay attention to the FOR/NEXT loops that go to 25 and to IF statements that test values against 25. Don't forget to change the DIM statements.

You might also be interested in keeping track of something more than the items we provided. It's relatively easy to add more items to keep track of. Just look at what the code is doing, and insert what you want.

Table 5-5. Trip Planner Routines

Lines	Function
100-780	Main program logic control code.
1000-1280	Data entry routine.
2000-2050	Solicit cost of gas and mile/gallon rating of the vehicle.
3000-3220	Calculate trip costs.
4000-4210	Print control routine.
5000-5250	Print a single day.
6000-6710	Change existing trip data.
7000-7100	Save trip data to a file.
8000-8100	Restore saved trip data from a file.
9000-9700	Delete some days from the trip.
10000-10290	Insert days into trip plan.
11000-11150	Initialize some system variables and explain the working of the program.
12000-12050	Get the current day.
13000-13040	Delay routine.

Table 5-6. Trip Planner Variables

Variable	Description
BUCKS	Total cost of the entire trip.
CHG	Set to one (1) to indicate a change to data that requires recalculation of totals.
DAY	The current day.
DAYS	The total number of days in this trip.
DEST$(25)	Destination name for each day.
DIST(25)	Distance to travel for each day.
DTOTAL(25)	Daily total costs for each day.
EATS(25)	Daily cost of food.
FILE$	Save filename.
FUN(25)	Daily other expenses.
GAS	Average cost of gasoline on the trip.
GCOST(25)	Daily cost of gasoline.
MOTEL(25)	Daily cost of lodging.
MPG	Average miles per gallon for the trip.
TOLLS(25)	Daily cost of tolls.
USER FUNCTION	
RD(X)	Rounds X to two digits past the decimal (nn.nn).

Listing 5-3. Trip Planner Program

```
100 REM TRIP PLANNER
110 DIM DEST$(25),DIST(25),TOLLS(25),MOTEL(25)
120 DIM EATS(25),FUN(25),DTOTAL(25),GCOST(25)
130 DEF RD(X)=INT(X*100+.5)/100
140 CALL CLEAR
150 PRINT TAB(7);"TRIP PLANNER"
160 PRINT : : : : : : : : : : : : :
170 FOR I=1 TO 200
180 NEXT I
190 CALL CLEAR
200 GOSUB 11000
210 CALL CLEAR
220 PRINT :"1. RESTORE SAVED PLAN":"2. SAVE TRIP PLAN":
    "3. ENTER TRIP DATA"
230 PRINT "4. EDIT TRIP DATA":"5. PRINT TRIP DATA":
    "6. DELETE A DAY"
240 PRINT "7. INSERT A DAY":"8. FINISHED"
250 PRINT : :
260 INPUT "YOUR CHOICE->":ANS
270 IF (ANS>0)*(ANS<9)THEN 300
280 CALL SOUND(-300,131,0,262,2,-2,1)
290 GOTO 210
300 ON ANS GOSUB 340,370,430,460,520,580,640,720
310 IF CHG=0 THEN 210
320 GOSUB 3000
330 GOTO 210
340 REM RESTORE DATA
350 GOSUB 8000
360 RETURN
370 REM SAVE DATA
380 IF DAYS<>0 THEN 410
390 PRINT : :"NO DATA TO SAVE."
400 RETURN
410 GOSUB 7000
420 RETURN
430 REM ENTER DATA
440 GOSUB 1000
450 RETURN
460 REM EDIT DATA
470 IF DAYS<>0 THEN 500
480 PRINT : :"NO DATA TO EDIT."
490 RETURN
500 GOSUB 6000
510 RETURN
520 REM PRINT DATA
530 IF DAYS<>0 THEN 560
540 PRINT : :"NO DATA TO PRINT."
550 RETURN
560 GOSUB 4000
570 RETURN
580 REM DELETE DATA
590 IF DAYS<>0 THEN 620
600 PRINT : :"NO DATA TO DELETE."
610 RETURN
620 GOSUB 9000
630 RETURN
640 REM DELETE DATA
```

Listing 5-3—cont. Trip Planner Program

```
650 IF DAYS<>25 THEN 700
660 CALL SOUND(-300,131,0,262,2,-2,1)
670 PRINT : :"NO ROOM TO INSERT DAY."
680 GOSUB 13000
690 RETURN
700 GOSUB 10000
710 RETURN
720 REM END OF RUN
730 STOP
1000 REM GET DATA
1010 SDAY=1
1020 IF DAYS=0 THEN 1060
1030 PRINT : :"BEGIN ENTERING DATA"
1040 GOSUB 12000
1050 SDAY=DAY
1060 CHG=1
1070 FOR DAY=SDAY TO 25
1080 GOSUB 1140
1090 IF DEST$(DAY)="X" THEN 1110
1100 NEXT DAY
1110 IF DAY<DAYS THEN 1130
1120 DAYS=DAY-1
1130 RETURN
1140 REM GET DAY DATA
1150 PRINT :"DESTINATION FOR DAY(X TO END)";DAY;"->";
1160 INPUT DEST$(DAY)
1170 IF DEST$(DAY)="X" THEN 1280
1180 PRINT :"DISTANCE TO "&DEST$(DAY)&"->";
1190 INPUT DIST(DAY)
1200 PRINT :"TOLLS FOR DAY";DAY;"->"
1210 INPUT TOLLS(DAY)
1220 PRINT :"LODGINGS FOR DAY";DAY;"->"
1230 INPUT MOTEL(DAY)
1240 PRINT :"FOOD  FOR DAY";DAY;"->"
1250 INPUT EATS(DAY)
1260 PRINT :"OTHER COSTS FOR DAY";DAY;"->"
1270 INPUT FUN(DAY)
1280 RETURN
2000 REM GET GAS COST AND MPG
2010 PRINT :"WHAT IS YOUR AVERAGE MILES"
2020 INPUT "PER GALLON->":MPG
2030 PRINT :"WHAT IS THE AVERAGE COST":"OF GASOLINE"
2040 INPUT "ON YOUR TRIP->":GAS
2050 RETURN
3000 REM CALCULATE COSTS
3010 IF MPG<>0 THEN 3040
3020 PRINT "I NEED SOME INFORMATION."
3030 GOSUB 2000
3040 CHG=0
3050 DIST(0)=0
3060 TOLLS(0)=0
3070 MOTEL(0)=0
3080 EATS(0)=0
3090 FUN(0)=0
3100 GCOST(0)=0
3110 FOR I=1 TO DAYS
3120 GCOST(I)=RD(DIST(I)/MPG*GAS)
```

Listing 5-3—cont. Trip Planner Program

```
3130 DTOTAL(I)=GCOST(I)+TOLLS(I)+MOTEL(I)+
     EATS(I)+FUN(I)
3140 DIST(0)=DIST(0)+DIST(I)
3150 TOLLS(0)=TOLLS(0)+TOLLS(I)
3160 MOTEL(0)=MOTEL(0)+MOTEL(I)
3170 EATS(0)=EATS(0)+EATS(I)
3180 FUN(0)=FUN(0)+FUN(I)
3190 GCOST(0)=GCOST(0)+GCOST(I)
3200 NEXT I
3210 BUCKS=RD(TOLLS(0)+MOTEL(0)+EATS(0)+
     FUN(0)+GCOST(0))
3220 RETURN
4000 REM PRINT CONTROL
4010 PRINT :"1. PRINT TOTALS ONLY":"2. PRINT ALL DAYS":
     "3. PRINT SELECTED DAYS"
4020 PRINT "4. EXIT": : :
4030 INPUT "YOUR CHOICE->":ANS
4040 IF (ANS<1)+(ANS>4)THEN 4000
4050 ON ANS GOTO 4060,4090,4150,4210
4060 DAY=0
4070 GOSUB 5000
4080 GOTO 4000
4090 FOR DAY=1 TO DAYS
4100 GOSUB 5000
4110 NEXT DAY
4120 DAY=0
4130 GOSUB 5000
4140 GOTO 4000
4150 PRINT :"PRINT WHICH DAY"
4160 INPUT "(0 TO END)->":DAY
4170 IF (DAY<0)+(DAY>25)THEN 4150
4180 IF DAY=0 THEN 4000
4190 GOSUB 5000
4200 GOTO 4150
4210 RETURN
5000 REM PRINT A DAY
5010 PRINT : :
5020 IF DAY=0 THEN 5090
5030 IF (DAY<=DAYS)*(DTOTAL(DAY)<>0)THEN 5070
5040 PRINT "NO DATA FOR DAY";DAY
5050 RETURN
5060 IF DAY=0 THEN 5090
5070 PRINT "DAY";DAY;TAB(10);DEST$(DAY)
5080 GOTO 5100
5090 PRINT "   ***TOTAL TRIP COSTS***"
5100 PRINT "MPG=";STR$(MPG);TAB(15);"GAS=$";STR$(GAS)
5110 PRINT "TOTAL TRIP COST=$";STR$(BUCKS)
5120 PRINT :"DISTANCE";TAB(15);DIST(DAY)
5130 PRINT "GASOLINE";TAB(15);"$";STR$(GCOST(DAY))
5140 PRINT "TOLLS";TAB(15);"$";STR$(TOLLS(DAY))
5150 PRINT "LODGINGS";TAB(15);"$";STR$(MOTEL(DAY))
5160 PRINT "FOOD";TAB(15);"$";STR$(EATS(DAY))
5170 PRINT "OTHER COSTS";TAB(15);"$";STR$(FUN(DAY))
5180 PRINT "--------------------------"
5190 IF DAY=0 THEN 5230
5200 PRINT :"DAY TOTAL";TAB(15);"$";STR$(DTOTAL(DAY))
5210 GOSUB 13000
```

Listing 5-3—cont. Trip Planner Program

```
5220 RETURN
5230 PRINT :"TRIP TOTAL";TAB(15);"$";STR$(BUCKS)
5240 GOSUB 13000
5250 RETURN
6000 REM EDIT EXISTING DATA
6010 CHG=0
6020 PRINT : :"EDIT DATA":"1. MILES PER GALLON":
     "2. COST OF GASOLINE"
6030 PRINT "3. DESTINATION":"4. DISTANCE":
     "5. COST OF LODGINGS":
6040 PRINT "6. COST OF FOOD":"7. TOLLS":
     "8. OTHER COSTS":"9. END"
6050 PRINT
6060 INPUT "CHANGE WHAT->":ANS
6070 IF (ANS<1)+(ANS>9)THEN 6020
6080 ON ANS GOTO 6090,6130,6170,6620,6260,6350,6440,
     6530,6710
6090 PRINT : :"MILES PER GALLON IS";MPG
6100 INPUT "ENTER NEW MPG->":MPG
6110 CHG=1
6120 GOTO 6020
6130 PRINT : :"COST OF GAS IS";GAS
6140 INPUT "ENTER NEW GAS COST->":GAS
6150 CHG=1
6160 GOTO 6020
6170 PRINT : :"CHANGE DESTINATION"
6180 GOSUB 12000
6190 IF DAY<=DAYS THEN 6220
6200 PRINT :"NO DATA FOR THAT DAY."
6210 GOTO 6020
6220 PRINT :"DESTINATION FOR DAY";DAY:"IS ";DEST$(DAY)
6230 INPUT "NEW DESTINATION->":DEST$(DAY)
6240 CHG=1
6250 GOTO 6020
6260 PRINT : :"CHANGE COST OF LODGING"
6270 GOSUB 12000
6280 IF DAY<=DAYS THEN 6310
6290 PRINT :"NO DATA FOR THAT DAY."
6300 GOTO 6020
6310 PRINT :"LODGINGS FOR DAY";DAY:"IS ";MOTEL(DAY)
6320 INPUT "NEW LODGINGS COST->":MOTEL(DAY)
6330 CHG=1
6340 GOTO 6020
6350 PRINT : :"CHANGE COST OF FOOD"
6360 GOSUB 12000
6370 IF DAY<=DAYS THEN 6400
6380 PRINT :"NO DATA FOR THAT DAY."
6390 GOTO 6020
6400 PRINT :"FOOD FOR DAY";DAY:"IS ";EATS(DAY)
6410 INPUT "NEW FOOD COST->":EATS(DAY)
6420 CHG=1
6430 GOTO 6020
6440 PRINT : :"CHANGE COST OF TOLLS"
6450 GOSUB 12000
6460 IF DAY<=DAYS THEN 6490
6470 PRINT :"NO DATA FOR THAT DAY."
6480 GOTO 6020
```

Listing 5-3—cont. Trip Planner Program

```
6490 PRINT :"TOLLS FOR DAY";DAY:"IS ";TOLLS(DAY)
6500 INPUT "NEW TOLLS->":TOLLS(DAY)
6510 CHG=1
6520 GOTO 6020
6530 PRINT : :"CHANGE OTHER COSTS"
6540 GOSUB 12000
6550 IF DAY<=DAYS THEN 6580
6560 PRINT :"NO DATA FOR THAT DAY."
6570 GOTO 6020
6580 PRINT :"OTHER COSTS FOR DAY";DAY:"ARE";FUN(DAY)
6590 INPUT "NEW OTHER COSTS->":FUN(DAY)
6600 CHG=1
6610 GOTO 6020
6620 PRINT : :"CHANGE DISTANCE"
6630 GOSUB 12000
6640 IF DAY<=DAYS THEN 6670
6650 PRINT :"NO DATA FOR THAT DAY."
6660 GOTO 6020
6670 PRINT :"DISTANCE FOR DAY";DAY:"IS ";DIST(DAY)
6680 INPUT "NEW DISTANCE->":DIST(DAY)
6690 CHG=1
6700 GOTO 6020
6710 RETURN
7000 REM SAVE TRIP DATA
7010 PRINT : : :
7020 INPUT "ENTER FILE NAME->":FILE$
7030 OPEN #10:FILE$,OUTPUT,INTERNAL
7040 PRINT #10:DAYS,MPG,GAS,BUCKS
7050 FOR I=0 TO DAYS
7060 PRINT #10:DEST$(I),DIST(I),TOLLS(I),MOTEL(I)
7070 PRINT #10:EATS(I),FUN(I),DTOTAL(I),GCOST(I)
7080 NEXT I
7090 CLOSE #10
7100 RETURN
8000 REM RESTORE TRIP DATA
8010 PRINT : : :
8020 INPUT "ENTER FILE NAME->":FILE$
8030 OPEN #10:FILE$,INPUT ,INTERNAL
8040 INPUT #10:DAYS,MPG,GAS,BUCKS
8050 FOR I=0 TO DAYS
8060 INPUT #10:DEST$(I),DIST(I),TOLLS(I),MOTEL(I)
8070 INPUT #10:EATS(I),FUN(I),DTOTAL(I),GCOST(I)
8080 NEXT I
8090 CLOSE #10
8100 RETURN
9000 REM DELETE TRIP DAYS
9010 PRINT : :"DELETE:": :"1. ONE DAY":
     "2. A RANGE OF DAYS":"3. EXIT"
9020 PRINT : :
9030 INPUT "YOUR CHOICE->":ANS
9040 IF (ANS<1)+(ANS>3)THEN 9000
9050 ON ANS GOTO 9060,9280,9700
9060 PRINT : :"DELETE TRIP DATA"
9070 GOSUB 12000
9080 IF DAY<=DAYS THEN 9110
9090 PRINT : :"NO DATA FOR THAT DAY."
9100 GOTO 9000
```

Listing 5-3—cont. Trip Planner Program

```
9110 DAYS=DAYS-1
9120 DELDAYS=1
9130 IF DAY+1<>DAYS THEN 9150
9140 GOTO 9580
9150 FOR I=DAY TO DAYS
9160 J=I+1
9170 DEST$(I)=DEST$(J)
9180 DIST(I)=DIST(J)
9190 TOLLS(I)=TOLLS(J)
9200 MOTEL(I)=MOTEL(J)
9210 EATS(I)=EATS(J)
9220 FUN(I)=FUN(J)
9230 DTOTAL(I)=DTOTAL(J)
9240 GCOST(I)=GCOST(J)
9250 NEXT I
9260 CHG=1
9270 GOTO 9580
9280 PRINT : :"DELETE TRIP DATA STARTING"
9290 INPUT "AT WHICH DAY->":SDAY
9300 IF SDAY<=DAYS THEN 9330
9310 PRINT : :"NO DATA FOR THAT DAY."
9320 GOTO 9000
9330 PRINT : :"DELETE TRIP DATA ENDING"
9340 INPUT "AT WHICH DAY->":EDAY
9350 IF EDAY<=DAYS THEN 9380
9360 PRINT : :"NO DATA FOR THAT DAY."
9370 GOTO 9000
9380 IF EDAY<>DAYS THEN 9430
9390 DELDAYS=DAYS-SDAY+1
9400 DAYS=SDAY-1
9410 CHG=1
9420 GOTO 9580
9430 DELDAYS=EDAY-SDAY+1
9440 DAYS=DAYS-DELDAYS
9450 FOR I=SDAY TO DAYS
9460 J=I+DELDAYS
9470 DEST$(I)=DEST$(J)
9480 DIST(I)=DIST(J)
9490 TOLLS(I)=TOLLS(J)
9500 MOTEL(I)=MOTEL(J)
9510 EATS(I)=EATS(J)
9520 FUN(I)=FUN(J)
9530 DTOTAL(I)=DTOTAL(J)
9540 GCOST(I)=GCOST(J)
9550 NEXT I
9560 CHG=1
9570 GOTO 9000
9580 FOR I=DAYS+1 TO DAYS+DELDAYS
9590 DEST$(I)=""
9600 DIST(I)=0
9610 DTOTAL(I)=0
9620 TOLLS(I)=0
9630 MOTEL(I)=0
9640 EATS(I)=0
9650 FUN(I)=0
9660 GCOST(I)=0
9670 NEXT I
```

Listing 5-3—cont. Trip Planner Program

```
9680 CHG=1
9690 GOTO 9000
9700 RETURN
10000 REM INSERT A DAY
10010 PRINT : :"INSERT DATA"
10020 GOSUB 12000
10030 IF DAY<=DAYS THEN 10060
10040 DAYS=DAY
10050 GOTO 10270
10060 DAYS=DAYS+1
10070 IF DAY=25 THEN 10270
10080 FOR I=DAYS TO DAY+1 STEP -1
10090 J=I-1
10100 DEST$(I)=DEST$(J)
10110 DIST(I)=DIST(J)
10120 TOLLS(I)=TOLLS(J)
10130 MOTEL(I)=MOTEL(J)
10140 EATS(I)=EATS(J)
10150 FUN(I)=FUN(J)
10160 DTOTAL(I)=DTOTAL(J)
10170 GCOST(I)=GCOST(J)
10180 NEXT I
10190 DEST$(DAY)=""
10200 DIST(DAY)=0
10210 TOLLS(DAY)=0
10220 MOTEL(DAY)=0
10230 EATS(DAY)=0
10240 FUN(DAY)=0
10250 GCOST(DAY)=0
10260 DTOTAL(DAY)=0
10270 GOSUB 1140
10280 CHG=1
10290 RETURN
11000 REM INIT AND TELL RULES
11010 DEST$(0)="TRIP TOTAL"
11020 DAYS=0
11030 PRINT "This is the trip planning":"program."
11040 PRINT :"You will be asked to enter":
      "the following data that"
11050 PRINT "applies to the entire":"trip:"
11060 PRINT :"miles/gallon you get":
      "average cost of gas"
11070 PRINT :"Then, for each day:"
11080 PRINT :"Destination":"Distance":"cost of tolls":
      "cost of food"
11090 PRINT "cost of lodging":"other costs"
11100 GOSUB 13000
11110 PRINT : :"You can then print reports":
      "for each day; total for the"
11120 PRINT "trip; change the data;":
      "store the data on tape or"
11130 PRINT "disk for later revision."
11140 GOSUB 13000
11150 RETURN
12000 REM GET THE DAY
12010 INPUT "FOR WHICH DAY(1-25)->";DAY
12020 IF (DAY>0)*(DAY<26)THEN 12050
```

Listing 5-3—cont. Trip Planner Program

```
12030 CALL SOUND(-300,131,0,262,2,-2,1)
12040 GOTO 12010
12050 RETURN
13000 REM DELAY FOR KEY STROKE
13010 PRINT :TAB(4);"HIT ANY KEY TO GO ON":
13020 CALL KEY(0,K,S)
13030 IF S=0 THEN 13020
13040 RETURN
```

6

Utilities

This chapter gives you these utility programs:

- CHARACTER EDITOR: Enters X's and .'s to create a new character. Get the hexadecimal string you need for a CALL to CHAR.
- GENERAL CONVERSION: Converts measurements between English and Metric units. Converts temperatures, volumes, areas, and angles.
- DEC/HEX/BIN CONVERSION: Converts numbers between the ever-popular binary, decimal, and hexadecimal number systems.
- OUTLINER: An outline organizer.

CHARACTER EDITOR PROGRAM

What it does:

The Character Editor program helps you to design custom characters for use in your own programs. It's also fun just to play with it.

On the TI-99/4A, characters on the screen are defined in an 8 × 8 dot (pixel) square. Two colors are assigned to a character—the *foreground color* and the *background color.* When a dot in the character square is "on," it shows the foreground color. "Off" dots show the background color.

In the Character Editor program, you are shown an 8 × 8 matrix of dots—the *pattern matrix.* These dots represent the background color. You then use simple commands to move the cursor over the 8 × 8 pattern matrix, placing an "X" over those dots that you want to appear in the foreground color.

Once you've defined the character pattern you want (with the X's

in the pattern matrix), you enter a U (update) command. The program then converts the pattern matrix data into a character definition string (for use with the CALL CHAR statement) and writes the character to the screen.

When you're satisfied with the character, you can ask for a print of the CALL CHAR pattern string required to produce it. We print this string in two-character chunks to make it easy for you to read. *Don't include the spaces when you use the string in a CALL CHAR statement in your own programs.*

For example, the heart-shaped player token used in the Tic-Tac-Toe program is printed as:

66 FF FF FF 7E 3C 18 18

To use this in a CALL CHAR statement, code:

```
CALL CHAR(128,"66FFFFFF7E3C1818")
```

The commands recognized by the Character Editor are:

- **C** allows you to change the color of the screen, the text, the pattern background, and the pattern foreground.
- **U** updates the character pattern displayed in the upper right corner of the screen to reflect the contents of the 8 × 8 pattern matrix.
- **BACK** (FCTN 8) ends the current character design and takes you to the point where you can print the character definition string.
- **A** is an on/off toggle that reverses the state of the automove function. If automove is on, it is turned off. If it is off, it is turned on. Initially, automove is off.
- **.** sets the corresponding dot in the character pattern to the background color.
- **X** sets the corresponding dot in the character pattern to the foreground color.
- **FCTN SEDX** are the directional arrow keys (FCTN shifted S, E, D, and X). They cause the cursor to move in the indicated direction and set the direction for the automove function.

Most of these commands are easy to understand. Once you start the program running, you will quickly learn what they do.

The *automove* feature, when turned on, causes the cursor to advance automatically in the direction of the last entered move (FCTN SEDX) command following each "." or "X" command. Use the A command to turn this feature on/off.

What it shows you:

The Character Editor program makes extensive use of direct output to the screen from TI BASIC. You can apply these techniques to simulate the DISPLAY AT and ACCEPT FROM Extended BASIC statements.

The color control in this program covers all the elements accessible from console BASIC. You can find code that changes the color of the screen, the text, and the foreground and background of the character you are designing.

What you can do:

Perhaps you would like to begin with an existing character pattern. You could add a routine that takes a pattern definition string (like that used in a CALL CHAR statement) and reverse converts it to the ones and zeros required in the CHAR pattern definition array.

If you're very ambitious, you can expand the program to allow design of more than one character at a time. Four characters, producing a 16 × 16 pattern matrix, fit nicely on the screen.

Table 6-1. Character Editor Routines

Lines	Function
100-290	Main program control code.
1000-1170	Initialize system variables and set default colors.
2000-2500	Write initial screen display.
3000-3090	Produce pattern definition string from CHAR array.
4000-4610	Read and act on keyboard commands.
5000-5760	Color change routine.
6000-6130	Print final results.
20000-20090	Direct output to the screen.
21000-21220	Direct read a two-digit number from the keyboard.
22000-22120	Direct read a Yes/No answer.

Table 6-2. Character Editor Variables

Variable	Description
ANS	The numeric answer from the numeric interface routine (21000 statement number range).
AUTOM	Indicates whether automove feature is on (1=on;0=off).
BCOLOR	Character background color.
CCOL	Current cursor column position in the pattern matrix.
CD(2)	Character to display in pattern definition matrix (CD(1)="." and CD(2)="X").
CHCOLOR	Text character color.
CROW	Current cursor row position in the pattern matrix.
D	Currently set direction for automove function. Set to the value of the arrow keys (**FCTN** SEDX).
HEXVAL	Contains the converted hexadecimal value representing four positions (bits) on the pattern image.
K	The last character read from the keyboard.
MAXI	The maximum row/column for the displayed pattern matrix.
MCOL	Message starting column for the direct output routine.
MLEN	Length of the message in the direct output routine.
MROW	Message row for the direct output routine.
MSG$	Message text written by the direct output routine.
OCOL	The column of the last character written by the direct output routine.
RTVAL	Return value; set to 99 when pattern design is complete.
SCRCOLOR	Screen color.
CAAR(8,8)	Contains the pattern, as ones for on bits, and zeros for off bits, of the character being designed.
CACOLOR	Foreground color of the character being designed.
CADEF$	The string containing the hexadecimal characters required by the CALL CHAR routine to produce the character you've designed.

Listing 6-1. Character Editor Program

```
100 OPTION BASE 1
110 REM
120 REM THIS PROGRAM CREATES A CHARACTER IMAGE
    INTERACTIVELY
130 REM
140 DIM CAAR(8,8)
150 CALL CLEAR
160 GOSUB 1000
170 CALL CLEAR
180 CALL HCHAR(4,22,128)
190 GOSUB 2000
200 PRINT "DO YOU WANT TO SEE"
210 INPUT "    THE PATTERN DATA?":Y$
220 IF (Y$="N")+(Y$="n") THEN 250
230 IF (Y$<>"Y")*(Y$<>"y") THEN 200
240 GOSUB 6000
250 INPUT "CREATE ANOTHER CHARACTER?":Y$
260 IF (Y$="N")+(Y$="n") THEN 290
```

Listing 6-1—cont. Character Editor Program

```
270 IF (Y$<>"Y")+(Y$<>"y") THEN 150
280 GOTO 250
290 STOP
1000 CALL CHAR(128,"0000000000000000")
1010 FOR I=1 TO 8
1020 FOR J=1 TO 8
1030 CAAR(I,J)=0
1040 NEXT J
1050 NEXT I
1060 SCRCOLOR=14
1070 CALL SCREEN(SCRCOLOR)
1080 CACOLOR=2
1090 CHCOLOR=16
1100 FOR C=1 TO 12
1110 CALL COLOR(C,CHCOLOR,1)
1120 NEXT C
1130 BCOLOR=SCRCOLOR
1140 CALL COLOR(13,CACOLOR,BCOLOR)
1150 CALL CLEAR
1160 CALL HCHAR(4,22,128)
1170 RETURN
2000 REM DISPLAY CHAR ROUTINE
2010 CD(1)=ASC(".")
2020 CD(2)=ASC("X")
2030 MAXI=8
2040 FOR I=1 TO MAXI
2050 FOR J=1 TO MAXI
2060 CALL HCHAR(I+2,J+3,CD(CAAR(I,J)+1))
2070 NEXT J
2080 NEXT I
2090 MSG$="COMMANDS"
2100 MROW=7
2110 MCOL=19
2120 GOSUB 20000
2130 MSG$="C=COLOR"
2140 MROW=MROW+1
2150 MCOL=19
2160 GOSUB 20000
2170 MSG$="U=UPDATE"
2180 MROW=MROW+1
2190 MCOL=19
2200 GOSUB 20000
2210 MSG$="BACK=DONE"
2220 MROW=MROW+1
2230 MCOL=19
2240 GOSUB 20000
2250 MSG$="A=AUTOMOVE"
2260 MROW=MROW+1
2270 MCOL=19
2280 GOSUB 20000
2290 MSG$=".=DOT OFF"
2300 MROW=MROW+1
2310 MCOL=19
2320 GOSUB 20000
2330 MSG$="X=DOT ON"
2340 MROW=MROW+1
2350 MCOL=19
```

Listing 6-1—cont. Character Editor Program

```
2360 GOSUB 20000
2370 MSG$="FCTN SEDX="
2380 MROW=MROW+2
2390 MCOL=19
2400 GOSUB 20000
2410 MSG$="DIRECTION"
2420 MROW=MROW+1
2430 MCOL=19
2440 GOSUB 20000
2450 CROW=3
2460 CCOL=2
2470 GOSUB 4000
2480 IF RTVAL=99 THEN 2500
2490 GOTO 2470
2500 RETURN
3000 REM BINARY TO HEX CONVERSION
3010 CADEF$=""
3020 FOR I=1 TO 8
3030 HEXVAL=CAAR(I,1)*8+CAAR(I,2)*4+CAAR(I,3)*2+
     CAAR(I,4)
3040 CADEF$=CADEF$&SEG$("0123456789ABCDEF",HEXVAL+1,1)
3050 HEXVAL=CAAR(I,5)*8+CAAR(I,6)*4+CAAR(I,7)*2+
     CAAR(I,8)
3060 CADEF$=CADEF$&SEG$("0123456789ABCDEF",HEXVAL+1,1)
3070 NEXT I
3080 CALL CHAR(128,CADEF$)
3090 RETURN
4000 REM READ KEYS AND ACT ON COMMANDS
4010 MAXI=9
4020 RTVAL=0
4030 D=9
4040 CALL KEY(0,K,S)
4050 CALL HCHAR(CROW,CCOL+2,30)
4060 IF CAAR(CROW-2,CCOL-1)=0 THEN 4090
4070 CALL HCHAR(CROW,CCOL+2,ASC("X"))
4080 GOTO 4100
4090 CALL HCHAR(CROW,CCOL+2,ASC("."))
4100 IF S=0 THEN 4040
4110 CALL SOUND(-150,1760,0)
4120 IF K<>ASC(".")THEN 4160
4130 CAAR(CROW-2,CCOL-1)=0
4140 CALL HCHAR(CROW,CCOL+2,ASC("."))
4150 GOTO 4350
4160 IF K<>ASC("X")THEN 4200
4170 CAAR(CROW-2,CCOL-1)=1
4180 CALL HCHAR(CROW,CCOL+2,ASC("X"))
4190 GOTO 4350
4200 IF K<>ASC("U")THEN 4230
4210 GOSUB 3000
4220 GOTO 4040
4230 IF K<>ASC("A")THEN 4260
4240 AUTOM=1-AUTOM
4250 GOTO 4040
4260 IF K<>ASC("C")THEN 4290
4270 GOSUB 5000
4280 GOTO 4040
4290 IF K<>15 THEN 4320
```

Listing 6-1—cont. Character Editor Program

```
4300 RTVAL=99
4310 RETURN
4320 IF (K<8)+(K>11)THEN 4520
4330 D=K
4340 GOTO 4360
4350 ON AUTOM+1 GOTO 4040,4360
4360 IF D<>8 THEN 4400
4370 IF CCOL=2 THEN 4040
4380 CCOL=CCOL-1
4390 GOTO 4040
4400 IF D<>9 THEN 4440
4410 IF CCOL=MAXI THEN 4040
4420 CCOL=CCOL+1
4430 GOTO 4040
4440 IF D<>11 THEN 4480
4450 IF CROW=3 THEN 4040
4460 CROW=CROW-1
4470 GOTO 4040
4480 IF D<>10 THEN 4520
4490 IF CROW=MAXI+1 THEN 4040
4500 CROW=CROW+1
4510 GOTO 4040
4520 CALL SOUND(-200,131,0,262,2,-2,1)
4530 MSG$="INVALID COMMAND!"
4540 MROW=22
4550 MCOL=3
4560 GOSUB 20000
4570 FOR S=1 TO 100
4580 NEXT S
4590 MSG$=""
4600 GOSUB 20000
4610 GOTO 4040
5000 REM SET COLORS
5010 MSG$="CHANGE SCREEN COLOR?"
5020 MROW=23
5030 MCOL=3
5040 GOSUB 20000
5050 GOSUB 22000
5060 MSG$=""
5070 GOSUB 20000
5080 ON YN GOTO 5090,5130
5090 C=SCRCOLOR
5100 GOSUB 5520
5110 CALL SCREEN(C)
5120 SCRCOLOR=C
5130 MSG$="CHANGE PATTERN COLOR?"
5140 MCOL=3
5150 GOSUB 20000
5160 GOSUB 22000
5170 MSG$=""
5180 GOSUB 20000
5190 ON YN GOTO 5200,5240
5200 C=CACOLOR
5210 GOSUB 5520
5220 CALL COLOR(13,C,BCOLOR)
5230 CACOLOR=C
5240 MSG$="CHANGE PAT BKGRD COLOR?"
```

Listing 6-1—cont. Character Editor Program

```
5250 MCOL=3
5260 GOSUB 20000
5270 GOSUB 22000
5280 MSG$=""
5290 GOSUB 20000
5300 ON YN GOTO 5310,5350
5310 C=BCOLOR
5320 GOSUB 5520
5330 CALL COLOR(13,CACOLOR,C)
5340 BCOLOR=C
5350 MSG$="CHANGE CHARACTER COLOR?"
5360 MROW=23
5370 MCOL=3
5380 GOSUB 20000
5390 GOSUB 22000
5400 ON YN GOTO 5410,5490
5410 MSG$=""
5420 GOSUB 20000
5430 C=CHCOLOR
5440 GOSUB 5520
5450 CHCOLOR=C
5460 FOR C=1 TO 12
5470 CALL COLOR(C,CHCOLOR,1)
5480 NEXT C
5490 MSG$=""
5500 GOSUB 20000
5510 RETURN
5520 MSG$="ENTER NEW COLOR."
5530 MCOL=3
5540 GOSUB 20000
5550 GOSUB 21000
5560 MSG$=""
5570 GOSUB 20000
5580 C=ANS
5590 IF (C>0)*(C<17)THEN 5630
5600 GOSUB 5690
5610 MROW=23
5620 GOTO 5520
5630 MROW=22
5640 MSG$=""
5650 GOSUB 20000
5660 MROW=23
5670 GOSUB 20000
5680 RETURN
5690 MSG$="COLORS RANGE FROM 1 TO 16."
5700 MCOL=3
5710 MROW=22
5720 GOSUB 20000
5730 CALL SOUND(-200,131,0,262,2,-2,1)
5740 FOR C=1 TO 100
5750 NEXT C
5760 RETURN
6000 REM PRINT THE CHAR DATA
6010 PRINT : :"DATA FOR PATTERN "
6020 PRINT "IN TWO CHARACTER PIECES ->"
6030 PRINT
6040 COUNT=1
```

Listing 6-1—cont. Character Editor Program

```
6050 FOR I=1 TO LEN(CADEF$)STEP 2
6060 PRINT SEG$(CADEF$,I,2)&" ";
6070 COUNT=COUNT+1
6080 IF COUNT<9 THEN 6110
6090 COUNT=1
6100 PRINT
6110 NEXT I
6120 PRINT
6130 RETURN
20000 REM PRINT MSG$ AT MROW STARTING IN MCOL
20010 MLEN=LEN(MSG$)
20020 IF MLEN=0 THEN 20080
20030 FOR OCOL=1 TO MLEN
20040 CALL HCHAR(MROW,OCOL+MCOL,ASC(SEG$(MSG$,OCOL,1)))
20050 NEXT OCOL
20060 MCOL=MCOL+OCOL
20070 RETURN
20080 CALL HCHAR(MROW,1,32,32)
20090 RETURN
21000 REM GET A UP TO TWO DIGIT NUMERIC ANSWER
21010 CALL KEY(0,ONE,S)
21020 IF S=0 THEN 21010
21030 CALL HCHAR(MROW,MCOL,ONE)
21040 ONE=ONE-48
21050 IF (ONE>=0)*(ONE<10)THEN 21080
21060 CALL SOUND(-200,131,0,262,2,-2,1)
21070 GOTO 21010
21080 MCOL=MCOL+1
21090 FOR K=1 TO 20
21100 NEXT K
21110 CALL KEY(0,TWO,S)
21120 IF S=0 THEN 21110
21130 IF TWO=13 THEN 21210
21140 CALL HCHAR(MROW,MCOL,TWO)
21150 TWO=TWO-48
21160 IF (TWO>=0)*(TWO<10)THEN 21190
21170 CALL SOUND(-200,131,0,262,2,-2,1)
21180 GOTO 21110
21190 ANS=ONE*10+TWO
21200 RETURN
21210 ANS=ONE
21220 RETURN
22000 REM GET YES/NO ANSWER
22010 CALL KEY(0,K,S)
22020 IF S=0 THEN 22010
22030 CALL HCHAR(MROW,MCOL,K)
22040 IF K<97 THEN 22060
22050 K=K-32
22060 YN=1
22070 IF (K=ASC("Y"))+(K=ASC("y")) THEN 22120
22080 YN=2
22090 IF (K=ASC("N"))+(K+ASC("n")) THEN 22120
22100 CALL SOUND(-200,131,0,262,2,-2,1)
22110 GOTO 22010
22120 RETURN
```

GENERAL CONVERSION PROGRAM

What it does:

This program performs a wide variety of conversions from units of one system to those of another. The program defines five classes of conversion:

- Distances and areas
- Volumes
- Weights
- Temperatures
- Angles

Within each of these broad conversion classes, there are detailed menus presented from which you choose the exact conversion you want performed.

The advantage of collecting so many conversions into one program is that once you have it, you have only one place to look for the conversions you need.

Under the *distances and areas* conversions, you can choose to perform conversions between:

- Inches and centimeters
- Feet and centimeters
- Feet and meters
- Yards and meters
- Miles and kilometers
- Light-years and miles
- Light-years and parsecs
- Furlongs and kilometers
- Acres and sq yards
- Acres and hectares

The *volume conversions* available to you are:

- Teaspoons and cc's
- Tablespoons and cc's
- Cups and cc's
- Cups and liters
- Pints and liters
- Quarts and liters
- Gallons and liters

In converting *weights,* you can choose from:

- Ounces and grams
- Pounds and grams
- Pounds and kilograms
- Tons and kilograms

The program allows *temperature conversion* between:

- Fahrenheit and centigrade
- Fahrenheit and kelvin
- Centigrade and kelvin

Finally, you can do conversions between *radians* and *degrees.*

Note that all of these conversions can be made in either direction. That is, you can convert, for example, from feet to centimeters or from centimeters to feet.

What it shows you:

Despite its size, this program contains only four variables. Only one of those is actually used in calculating the answers. The other three support the interaction with the user of the program.

Its simple purpose and clean, modular structure make it easy for a novice programmer to understand, change, and add to this program.

What you can do:

There are quite a few conversions already in this program, but perhaps you have some that you would like to add. Either insert them into the proper class, or define a new class to put your own conversions into.

Maybe you feel you won't need some conversion included in the program. If so, take it out. We don't recommend changing the line numbers unless you have a printer.

Table 6-3. General Conversion Routines

Lines	Function
100-220	Main control loop.
1000-2630	Distance and area conversions.
4000-5160	Volume conversions.
6000-6700	Weight conversions.
8000-8600	Temperature conversions.
10000-10160	Degree/radian conversion.
11000-11080	Yes/No question routine.

Table 6-4. General Conversion Variables

Variable	Description
ANS	Used for all computations.
MSG$	Message for the Yes/No routine.
Y$	Yes/No answer from keyboard.
YN	Set to one (1) for Yes; set to two (2) for No.

Listing 6-2. General Conversion Program

```
100 REM CONVERSION PROGRAM
110 CALL CLEAR
120 PRINT : : : :"1. Distances and areas":"2. Volumes":
    "3. Weights"
130 PRINT "4. Temperatures":"5. Angles":"6. END"
140 PRINT : : : : : : :
150 INPUT "Your choice->":ANS
160 IF (ANS>0)*(ANS<7)THEN 190
170 CALL SOUND(300,131,2,262,4,-2,3)
180 GOTO 110
190 CALL CLEAR
200 ON ANS GOSUB 1000,4000,6000,8000,10000,220
210 GOTO 120
220 STOP
1000 REM DISTANCES
1010 PRINT : : :"1. Inches and centimeters":
     "2. Feet and centimeters"
1020 PRINT "3. Feet and meters":"4. Yards and meters"
1030 PRINT "5. Miles and kilometers":
     "6. Light-years and miles":
1040 PRINT "7. Light-years and parsecs":
     "8. Furlongs and kilometers"
1050 PRINT "9. Acres and sq yards":
     "10. Acres and hectares"
1060 PRINT "11. END": : : : : :
1070 INPUT "Your choice->":ANS
1080 IF (ANS>0)*(ANS<12)THEN 1110
1090 CALL SOUND(300,131,2,262,4,-2,3)
1100 GOTO 1010
1110 IF ANS<>11 THEN 1130
1120 RETURN
1130 ON ANS GOTO 1140,1290,1440,1590,1740,1890,2040,
     2190,2340,2490
1140 REM INCHES/CENTIMETERS
1150 PRINT : :"CONVERT INCHES"
1160 MSG$="   TO CENTIMETERS(Y/N)?"
1170 GOSUB 11000
1180 IF YN=2 THEN 1240
1190 INPUT "ENTER INCHES(0 TO END)->":ANS
1200 IF ANS=0 THEN 1010
1210 ANS=2.54*ANS
1220 PRINT : : :ANS;" CENTIMETERS"
1230 GOTO 1190
1240 INPUT "ENTER CENTIMETERS(0 TO END)->":ANS
1250 IF ANS=0 THEN 1010
1260 ANS=ANS*.393701
1270 PRINT : : :ANS;" INCHES"
1280 GOTO 1240
```

Listing 6-2—cont. General Conversion Program

```
1290 REM FEET/CENTIMETERS
1300 PRINT : :"CONVERT FEET"
1310 MSG$="   TO CENTIMETERS(Y/N)?"
1320 GOSUB 11000
1330 IF YN=2 THEN 1390
1340 INPUT "ENTER FEET(0 TO END)->":ANS
1350 IF ANS=0 THEN 1010
1360 ANS=ANS*30.48
1370 PRINT : : :ANS;" CENTIMETERS"
1380 GOTO 1340
1390 INPUT "ENTER CENTIMETERS(0 TO END)->":ANS
1400 IF ANS=0 THEN 1010
1410 ANS=ANS/30.48
1420 PRINT : : :ANS;" FEET"
1430 GOTO 1390
1440 REM FEET/METERS
1450 PRINT : :"CONVERT FEET"
1460 MSG$="   TO METERS(Y/N)?"
1470 GOSUB 11000
1480 IF YN=2 THEN 1540
1490 INPUT "ENTER FEET(0 TO END)->":ANS
1500 IF ANS=0 THEN 1010
1510 ANS=ANS*3.28084
1520 PRINT : : :ANS;" METERS"
1530 GOTO 1490
1540 INPUT "ENTER METERS(0 TO END)->":ANS
1550 IF ANS=0 THEN 1010
1560 ANS=ANS*.3048
1570 PRINT : : :ANS;" FEET"
1580 GOTO 1540
1590 REM YARDS/METERS
1600 PRINT : :"CONVERT YARDS"
1610 MSG$="   TO METERS(Y/N)?"
1620 GOSUB 11000
1630 IF YN=2 THEN 1690
1640 INPUT "ENTER YARDS(0 TO END)->":ANS
1650 IF ANS=0 THEN 1010
1660 ANS=ANS*1.0936133
1670 PRINT : : :ANS;" METERS"
1680 GOTO 1640
1690 INPUT "ENTER METERS(0 TO END)->":ANS
1700 IF ANS=0 THEN 1010
1710 ANS=ANS*.9144
1720 PRINT : : :ANS;" YARDS"
1730 GOTO 1690
1740 REM MILES/KILOMETERS
1750 PRINT : :"CONVERT MILES"
1760 MSG$="   TO KILOMETERS(Y/N)?"
1770 GOSUB 11000
1780 IF YN=2 THEN 1840
1790 INPUT "ENTER MILES(0 TO END)->":ANS
1800 IF ANS=0 THEN 1010
1810 ANS=ANS*1.609344
1820 PRINT : : :ANS;" KILOMETERS"
1830 GOTO 1790
1840 INPUT "ENTER KILOMETERS(0 TO END)->":ANS
1850 IF ANS=0 THEN 1010
```

Listing 6-2—cont. General Conversion Program

```
1860 ANS=ANS/1.609344
1870 PRINT : : :ANS;" MILES"
1880 GOTO 1840
1890 REM LIGHTYEARS/MILES
1900 PRINT : :"CONVERT MILES"
1910 MSG$="   TO LIGHTYEARS(Y/N)?"
1920 GOSUB 11000
1930 IF YN=2 THEN 1990
1940 INPUT "ENTER MILES(0 TO END)->":ANS
1950 IF ANS=0 THEN 1010
1960 ANS=ANS/5.878E12
1970 PRINT : : :ANS;" LIGHTYEARS"
1980 GOTO 1940
1990 INPUT "ENTER LIGHTYEARS(0 TO END)->":ANS
2000 IF ANS=0 THEN 1010
2010 ANS=ANS*5.878E12
2020 PRINT : : :ANS;" MILES"
2030 GOTO 1990
2040 REM LIGHTYEARS/PARSECS
2050 PRINT : :"CONVERT LIGHTYEARS"
2060 MSG$="   TO PARSECS(Y/N)?"
2070 GOSUB 11000
2080 IF YN=2 THEN 2140
2090 INPUT "ENTER LIGHTYEARS(0 TO END)->":ANS
2100 IF ANS=0 THEN 1010
2110 ANS=ANS/3.2615
2120 PRINT : : :ANS;" PARSECS"
2130 GOTO 2090
2140 INPUT "ENTER PARSECS(0 TO END)->":ANS
2150 IF ANS=0 THEN 1010
2160 ANS=ANS*3.2615
2170 PRINT : : :ANS;" LIGHTYEARS"
2180 GOTO 2140
2190 REM FURLONG/KILOMETERS
2200 PRINT : :"CONVERT FURLONGS"
2210 MSG$="   TO KILOMETERSS(Y/N)?"
2220 GOSUB 11000
2230 IF YN=2 THEN 2290
2240 INPUT "ENTER FURLONGS(0 TO END)->":ANS
2250 IF ANS=0 THEN 1010
2260 ANS=ANS*.2011675
2270 PRINT : : :ANS;" KILOMETERS"
2280 GOTO 2240
2290 INPUT "ENTER KILOMETERS(0 TO END)->":ANS
2300 IF ANS=0 THEN 1010
2310 ANS=ANS/.2011675
2320 PRINT : : :ANS;" FURLONGS"
2330 GOTO 2140
2340 REM ACRES/SQ YARDS
2350 PRINT : :"CONVERT ACRES"
2360 MSG$="   TO SQ YARDS(Y/N)?"
2370 GOSUB 11000
2380 IF YN=2 THEN 2490
2390 INPUT "ENTER ACRES(0 TO END)->":ANS
2400 IF ANS=0 THEN 1010
2410 ANS=ANS*4840
2420 PRINT : : :ANS;" SQ YARDS"
```

Listing 6-2—cont. General Conversion Program

```
2430 GOTO 2390
2440 INPUT "ENTER SQ YARDS(0 TO END)->":ANS
2450 IF ANS=0 THEN 1010
2460 ANS=ANS/4840
2470 PRINT : : :ANS;" ACRES"
2480 GOTO 2440
2490 REM ACRES/HECTARES
2500 PRINT : :"CONVERT ACRES"
2510 MSG$="   TO HECTARES(Y/N)?"
2520 GOSUB 11000
2530 IF YN=2 THEN 2590
2540 INPUT "ENTER ACRES(0 TO END)->":ANS
2550 IF ANS=0 THEN 1010
2560 ANS=ANS*.40469
2570 PRINT : : :ANS;" HECTARES"
2580 GOTO 2540
2590 INPUT "ENTER HECTARES(0 TO END)->":ANS
2600 IF ANS=0 THEN 1010
2610 ANS=ANS*2.471
2620 PRINT : : :ANS;" ACRES"
2630 GOTO 2590
4000 REM VOLUMES
4010 PRINT : : :"1. Teaspoons and cc's":
     "2. Tablespoons and cc's"
4020 PRINT "3. Cups and cc's":"4. Cups and liters"
4030 PRINT "5. Pints and liters":
     "6. Quarts and liters":
4040 PRINT "7. Gallons and liters":"8. END": : : : : :
4050 INPUT "Your choice->":ANS
4060 IF (ANS>0)*(ANS<9)THEN 4090
4070 CALL SOUND(300,131,2,262,4,-2,3)
4080 GOTO 4010
4090 IF ANS<>8 THEN 4110
4100 RETURN
4110 ON ANS GOTO 4120,4270,4420,4570,4720,4870,5020
4120 REM TEASPOONS/CUBIC CENTIMETERS
4130 PRINT : :"CONVERT TEASPOONS TO CUBIC"
4140 MSG$="   CENTIMETERS(Y/N)?"
4150 GOSUB 11000
4160 IF YN=2 THEN 4220
4170 INPUT "ENTER TEASPOONS(0 TO END)->":ANS
4180 IF ANS=0 THEN 4010
4190 ANS=ANS*4.929
4200 PRINT : : :ANS;" CC's"
4210 GOTO 4170
4220 INPUT "ENTER CC's(0 TO END)->":ANS
4230 IF ANS=0 THEN 4010
4240 ANS=ANS/4.929
4250 PRINT : : :ANS;" TEASPOONS"
4260 GOTO 4220
4270 REM TABLESPOONS/CUBIC CENTIMETERS
4280 PRINT : :"CONVERT TABLESPOONS TO"
4290 MSG$="  CUBIC CENTIMETERS(Y/N)?"
4300 GOSUB 11000
4310 IF YN=2 THEN 4370
4320 INPUT "ENTER TABLESPOONS(0 TO END)->":ANS
4330 IF ANS=0 THEN 4010
4340 ANS=ANS*14.788
```

Listing 6-2—cont. General Conversion Program

```
4350 PRINT : : :ANS;" CC's"
4360 GOTO 4320
4370 INPUT "ENTER CC's(0 TO END)->":ANS
4380 IF ANS=0 THEN 4010
4390 ANS=ANS/14.788
4400 PRINT : : :ANS;" TABLESPOONS"
4410 GOTO 4370
4420 REM CUPS/CUBIC CENTIMETERS
4430 PRINT : :"CONVERT CUPS TO CUBIC"
4440 MSG$="   CENTIMETERS(Y/N)?"
4450 GOSUB 11000
4460 IF YN=2 THEN 4520
4470 INPUT "ENTER CUPS(0 TO END)->":ANS
4480 IF ANS=0 THEN 4010
4490 ANS=ANS*236.6
4500 PRINT : : :ANS;" CC's"
4510 GOTO 4470
4520 INPUT "ENTER CC's(0 TO END)->":ANS
4530 IF ANS=0 THEN 4010
4540 ANS=ANS/236.6
4550 PRINT : : :ANS;" CUPS"
4560 GOTO 4520
4570 REM CUPS/LITERS
4580 PRINT : :"CONVERT CUPS"
4590 MSG$="   TO LITERS(Y/N)?"
4600 GOSUB 11000
4610 IF YN=2 THEN 4670
4620 INPUT "ENTER CUPS(0 TO END)->":ANS
4630 IF ANS=0 THEN 4010
4640 ANS=ANS*.2366
4650 PRINT : : :ANS;" LITERS"
4660 GOTO 4620
4670 INPUT "ENTER LITERS(0 TO END)->":ANS
4680 IF ANS=0 THEN 4010
4690 ANS=ANS/.2366
4700 PRINT : : :ANS;" CUPS"
4710 GOTO 4670
4720 REM PINTS/LITERS
4730 PRINT : :"CONVERT PINTS"
4740 MSG$="   TO LITERS(Y/N)?"
4750 GOSUB 11000
4760 IF YN=2 THEN 4820
4770 INPUT "ENTER PINTS(0 TO END)->":ANS
4780 IF ANS=0 THEN 4010
4790 ANS=ANS*.4732
4800 PRINT : : :ANS;" LITERS"
4810 GOTO 4770
4820 INPUT "ENTER LITERS(0 TO END)->":ANS
4830 IF ANS=0 THEN 4010
4840 ANS=ANS/.4732
4850 PRINT : : :ANS;" PINTS"
4860 GOTO 4820
4870 REM QUARTS/LITERS
4880 PRINT : :"CONVERT QUARTS"
4890 MSG$="   TO LITERS(Y/N)?"
4900 GOSUB 11000
4910 IF YN=2 THEN 4970
```

Listing 6-2—cont. General Conversion Program

```
4920 INPUT "ENTER QUARTS(0 TO END)->":ANS
4930 IF ANS=0 THEN 4010
4940 ANS=ANS*.9463
4950 PRINT : : :ANS;" LITERS"
4960 GOTO 4920
4970 INPUT "ENTER LITERS(0 TO END)->":ANS
4980 IF ANS=0 THEN 4010
4990 ANS=ANS/.9463
5000 PRINT : : :ANS;" QUARTS"
5010 GOTO 4970
5020 REM GALLONS/LITERS
5030 PRINT : :"CONVERT GALLONS"
5040 MSG$="   TO LITERS(Y/N)?"
5050 GOSUB 11000
5060 IF YN=2 THEN 5120
5070 INPUT "ENTER GALLONS(0 TO END)->":ANS
5080 IF ANS=0 THEN 4010
5090 ANS=ANS*3.785
5100 PRINT : : :ANS;" LITERS"
5110 GOTO 5070
5120 INPUT "ENTER LITERS(0 TO END)->":ANS
5130 IF ANS=0 THEN 4010
5140 ANS=ANS/3.785
5150 PRINT : : :ANS;" GALLONS"
5160 GOTO 5120
6000 REM WEIGHT
6010 PRINT : : :"1. Ounces and grams":
     "2. Pounds and grams"
6020 PRINT "3. Pounds and kilograms":
     "4. Tons and kilograms"
6030 PRINT "5. END": : : : : : :
6040 INPUT "Your choice->":ANS
6050 IF (ANS>0)*(ANS<6)THEN 6080
6060 CALL SOUND(300,131,2,262,4,-2,3)
6070 GOTO 6010
6080 IF ANS<>5 THEN 6100
6090 RETURN
6100 ON ANS GOTO 6110,6260,6410,6560
6110 REM OUNCES/GRAMS
6120 PRINT : :"CONVERT OUNCES"
6130 MSG$="   TO GRAMS(Y/N)?"
6140 GOSUB 11000
6150 IF YN=2 THEN 6210
6160 INPUT "ENTER OUNCES(0 TO END)->":ANS
6170 IF ANS=0 THEN 6010
6180 ANS=ANS*28.349523
6190 PRINT : : :ANS;" GRAMS"
6200 GOTO 6160
6210 INPUT "ENTER GRAMS(0 TO END)->":ANS
6220 IF ANS=0 THEN 6010
6230 ANS=ANS/28.349523
6240 PRINT : : :ANS;" OUNCES"
6250 GOTO 6210
6260 REM POUNDS/GRAMS
6270 PRINT : :"CONVERT POUNDS"
6280 MSG$="   TO GRAMS(Y/N)?"
6290 GOSUB 11000
```

Listing 6-2—cont. General Conversion Program

```
6300 IF YN=2 THEN 6360
6310 INPUT "ENTER POUNDS(0 TO END)->":ANS
6320 IF ANS=0 THEN 6010
6330 ANS=ANS*453.59237
6340 PRINT : : :ANS;" GRAMS"
6350 GOTO 6310
6360 INPUT "ENTER GRAMS(0 TO END)->":ANS
6370 IF ANS=0 THEN 6010
6380 ANS=ANS/453.59237
6390 PRINT : : :ANS;" POUNDS"
6400 GOTO 6360
6410 REM POUNDS/KILOGRAMS
6420 PRINT : :"CONVERT POUNDS"
6430 MSG$="   TO KILOGRAMS(Y/N)?"
6440 GOSUB 11000
6450 IF YN=2 THEN 6510
6460 INPUT "ENTER POUNDS(0 TO END)->":ANS
6470 IF ANS=0 THEN 6010
6480 ANS=ANS*.453592
6490 PRINT : : :ANS;" KILOGRAMS"
6500 GOTO 6460
6510 INPUT "ENTER KILOGRAMS(0 TO END)->":ANS
6520 IF ANS=0 THEN 6010
6530 ANS=ANS/.453592
6540 PRINT : : :ANS;" POUNDS"
6550 GOTO 6510
6560 REM TONS/KILOGRAMS
6570 PRINT : :"CONVERT TONS"
6580 MSG$="   TO KILOGRAMS(Y/N)?"
6590 GOSUB 11000
6600 IF YN=2 THEN 6660
6610 INPUT "ENTER TONS(0 TO END)->":ANS
6620 IF ANS=0 THEN 6010
6630 ANS=ANS*907.18474
6640 PRINT : : :ANS;" KILOGRAMS"
6650 GOTO 6610
6660 INPUT "ENTER KILOGRAMS(0 TO END)->":ANS
6670 IF ANS=0 THEN 6010
6680 ANS=ANS/907.18474
6690 PRINT : : :ANS;" TONS"
6700 GOTO 6660
8000 REM TEMPERATURES
8010 PRINT : : :"1. Fahrenheit and Centigrade":
     "2. Fahrenheit and Kelvin"
8020 PRINT "3. Centigrade and Kelvin":
     "4. END": : : : : : :
8030 INPUT "Your choice->":ANS
8040 IF (ANS>0)*(ANS<5)THEN 8070
8050 CALL SOUND(300,131,2,262,4,-2,3)
8060 GOTO 8010
8070 IF ANS<>4 THEN 8090
8080 RETURN
8090 ON ANS GOTO 8100,8270,8440
8100 REM FARENHEIT/CENTIGRADE
8110 PRINT : :"CONVERT FAHRENHEIT"
8120 MSG$="   TO CENTIGRADE(Y/N)?"
8130 GOSUB 11000
```

Listing 6-2—cont. General Conversion Program

```
8140 IF YN=2 THEN 8210
8150 PRINT "ENTER FARENHEIT DEGREES"
8160 INPUT "(-999 TO END)->":ANS
8170 IF ANS=-999 THEN 8010
8180 ANS=(ANS-32)*.555555
8190 PRINT : : :ANS;" DEGREES C"
8200 GOTO 8150
8210 PRINT "ENTER CENTIGRADE DEGREES"
8220 INPUT "(-999 TO END)->":ANS
8230 IF ANS=-999 THEN 8010
8240 ANS=1.8*ANS+32
8250 PRINT : : :ANS;" DEGREES F"
8260 GOTO 8210
8270 REM KELVIN/FAHRENHEIT
8280 PRINT : :"CONVERT FAHRENHEIT"
8290 MSG$="   TO KELVIN(Y/N)?"
8300 GOSUB 11000
8310 IF YN=2 THEN 8380
8320 PRINT "ENTER FARENHEIT DEGREES"
8330 INPUT "(-999 TO END)->":ANS
8340 IF ANS=-999 THEN 8010
8350 ANS=(ANS+459.69)*.555555
8360 PRINT : : :ANS;" DEGREES K"
8370 GOTO 8320
8380 PRINT "ENTER KELVIN DEGREES"
8390 INPUT "(-999 TO END)->":ANS
8400 IF ANS=-999 THEN 8010
8410 ANS=ANS*1.8-459.69
8420 PRINT : : :ANS;" DEGREES F"
8430 GOTO 8380
8440 REM KELVIN/CENTIGRADE
8450 PRINT : :"CONVERT CENTIGRADE"
8460 MSG$="   TO KELVIN(Y/N)?"
8470 GOSUB 11000
8480 IF YN=2 THEN 8550
8490 PRINT "ENTER CENTIGRADE DEGREES"
8500 INPUT "(-999 TO END)->":ANS
8510 IF ANS=-999 THEN 8010
8520 ANS=ANS+273.16
8530 PRINT : : :ANS;" DEGREES K"
8540 GOTO 8490
8550 PRINT "ENTER KELVIN DEGREES"
8560 INPUT "(-999 TO END)->":ANS
8570 IF ANS=-999 THEN 8010
8580 ANS=ANS-273.16
8590 PRINT : : :ANS;" DEGREES C"
8600 GOTO 8550
10000 REM DEGREES
10010 REM DEGREES/RADIANS
10020 PRINT : :"CONVERT DEGREES"
10030 MSG$="   TO RADIANS(Y/N)?"
10040 GOSUB 11000
10050 IF YN=2 THEN 10110
10060 INPUT "ENTER DEGREES(0 TO END)->":ANS
10070 IF ANS=0 THEN 10160
10080 ANS=ANS*.01745
10090 PRINT : : :ANS;" RADIANS"
```

Listing 6-2—cont. General Conversion Program

```
10100 GOTO 10060
10110 INPUT "ENTER RADIANS(0 TO END)->":ANS
10120 IF ANS=0 THEN 10160
10130 ANS=ANS*57.295779
10140 PRINT : : :ANS;" DEGREES"
10150 GOTO 10110
10160 RETURN
11000 REM YES/NO
11010 INPUT MSG$:Y$
11020 YN=1
11030 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")
      THEN 11080
11040 YN=2
11050 IF (SEG$(Y$,1,1)="N")+(SEG$(Y$,1,1)="n")
      THEN 11080
11060 CALL SOUND(300,131,2,262,4,-2,3)
11070 GOTO 11000
11080 RETURN
```

DEC/HEX/BIN CONVERSION PROGRAM

What it does:

This program allows you to enter a binary, hexadecimal, or decimal number. It then prints that number on all three formats. When you enter a number, you must tell what format it's in:

- Enter *decimal numbers* in the normal way. For example, 123.
- Enter *hexadecimal numbers* by ending the number with the letter H. For example, FA5BH.
- Enter *binary numbers* by ending the number with the letter B. For example, 1101101B.

That's all there is to it. Whatever format number you enter, results are printed to your screen in all three formats.

What it shows you:

If you're interested in methods for converting between the various number bases, routines included in this program show how to do it.

What you can do:

We didn't include octal (base 8) in this conversion program because it's not commonly encountered in systems based on an 8-bit byte, like the TI-99/4A. If you have need for an octal conversion, it is fairly easy to add to this program.

Table 6-5. Dec/Hex/Bin Conversion Routines

Lines	Function
100-300	Print rules and initialize tables.
310-410	Main control loop; get input and print results to the screen.
1000-1220	Check data type of the input variable and call the appropriate conversion routines.
2000-2180	Decimal to hexadecimal conversion.
3000-3130	Hexadecimal to binary conversion.
4000-4150	Hexadecimal to decimal conversion.
5000-5140	Binary to decimal conversion.
6000-6350	Conversion table data.

Table 6-6. Dec/Hex/Bin Conversion Variables

Variable	Description
A$	The value to be converted.
BAD	When not zero, indicates a conversion error (usually an illegal character in the input value).
BIN$	The binary result.
BINVAL$(15)	Table of four bit binary values whose position in this array corresponds to their decimal values.
DEC	The decimal result.
HEX$	The hexadecimal result.
HEXVAL$(255)	Table of hexadecimal values whose position in this array corresponds to their decimal value.

Listing 6-3. Dec/Hex/Bin Conversion Program

```
100 REM THIS PROGRAM COMPUTES THE
110 REM HEX, BINARY, AND DECIMAL
120 REM VALUES OF ANY NUMBER ENTERED.
130 REM
140 DIM BINVAL$(15),HEXVAL$(255)
150 CALL CLEAR
160 PRINT "DEC/HEX/BINARY CONVERTER": : :
170 PRINT "ENTER DECIMAL NUMBERS IN":"THE USUAL WAY.":
    "FOR EXAMPLE, 12345"
180 PRINT : :"ENTER HEXADECIMAL NUMBERS":
    "BY ENDING THE NUMBER WITH":"AN H."
190 PRINT "FOR EXAMPLE, FE45H"
200 PRINT : :"ENTER BINARY NUMBERS BY":
    "ENDING THE NUMBER WITH B."
210 PRINT "FOR EXAMPLE, 11010B"
220 REM
230 REM LOAD THE LOOKUP TABLES
240 PRINT : :"LOADING TABLES ...": :
250 FOR I=0 TO 15
260 READ BINVAL$(I)
270 NEXT I
280 FOR I=0 TO 255
290 READ HEXVAL$(I)
```

Listing 6-3—cont. Dec/Hex/Bin Conversion Program

```
300 NEXT I
310 PRINT : :"ENTER THE VALUE"
320 INPUT "  (X TO END)->":A$
330 IF A$="X" THEN 410
340 BAD=0
350 GOSUB 1000
360 IF BAD THEN 310
370 PRINT : :"DECIMAL=";DEC
380 PRINT :"HEXADECIMAL=";HEX$
390 PRINT :"BINARY=";BIN$
400 GOTO 310
410 STOP
1000 REM CHECK THE DATA TYPE
1010 REM THEN GOSUB THE RIGHT
1020 REM ROUTINES
1030 LAST$=SEG$(A$,LEN(A$),1)
1040 IF LAST$="B" THEN 1130
1050 IF LAST$="H" THEN 1180
1060 GOSUB 2120
1070 IF BAD THEN 1120
1080 DEC=VAL(A$)
1090 GOSUB 2000
1100 IF BAD THEN 1120
1110 GOSUB 3000
1120 RETURN
1130 BIN$=SEG$(A$,1,LEN(A$)-1)
1140 GOSUB 5000
1150 IF BAD THEN 1170
1160 GOSUB 2000
1170 RETURN
1180 HEX$=SEG$(A$,1,LEN(A$)-1)
1190 GOSUB 4000
1200 IF BAD THEN 1220
1210 GOSUB 3000
1220 RETURN
2000 REM DECIMAL TO HEX CONVERSION
2010 V=DEC
2020 HEX$=""
2030 X=INT(V/256)
2040 B=V-X*256
2050 HEX$=HEXVAL$(B)&HEX$
2060 IF X=0 THEN 2090
2070 V=X
2080 GOTO 2030
2090 IF SEG$(HEX$,1,1)<>"0" THEN 2110
2100 HEX$=SEG$(HEX$,2,255)
2110 RETURN
2120 FOR I=1 TO LEN(A$)
2130 IF POS("0123456789",SEG$(A$,I,1),1)=0 THEN 2160
2140 NEXT I
2150 RETURN
2160 PRINT : :"ILLEGAL CHAR (";SEG$(A$,I,1);") IN":
     "DECIMAL VALUE ";A$
2170 BAD=-1
2180 RETURN
3000 REM HEX TO BINARY CONVERSION
3010 BIN$=""
```

Listing 6-3—cont. Dec/Hex/Bin Conversion Program

```
3020 FOR I=1 TO LEN(HEX$)
3030 C=ASC(SEG$(HEX$,I,1))
3040 C=C-48
3050 IF C<10 THEN 3070
3060 C=C-7
3070 IF (C<0)+(C>15)THEN 3110
3080 BIN$=BIN$&BINVAL$(C)
3090 NEXT I
3100 RETURN
3110 PRINT : :"ILLEGAL CHAR (";SEG$(HEX$,I,1);") IN":
     "HEX VALUE ";HEX$
3120 BAD=-1
3130 RETURN
4000 REM HEX TO DECIMAL CONVERSION
4010 DEC=0
4020 L=LEN(HEX$)-1
4030 FOR I=1 TO LEN(HEX$)
4040 C=ASC(SEG$(HEX$,I,1))
4050 C=C-48
4060 IF C<10 THEN 4080
4070 C=C-7
4080 IF (C<0)+(C>15)THEN 4130
4090 DEC=DEC+C*(16^L)
4100 L=L-1
4110 NEXT I
4120 RETURN
4130 PRINT : :"ILLEGAL CHAR (";SEG$(HEX$,I,1);") IN":
     "HEX VALUE ";HEX$
4140 BAD=-1
4150 RETURN
5000 REM BIN TO DECIMAL CONVERSION
5010 DEC=0
5020 L=LEN(BIN$)-1
5030 FOR I=1 TO LEN(BIN$)
5040 C=ASC(SEG$(BIN$,I,1))
5050 C=C-48
5060 IF (C<0)+(C>1)THEN 5120
5070 IF C=0 THEN 5090
5080 DEC=DEC+2^L
5090 L=L-1
5100 NEXT I
5110 RETURN
5120 PRINT : :"ILLEGAL CHAR (";SEG$(BIN$,I,1);") IN":
     "BINARY VALUE ";BIN$
5130 BAD=-1
5140 RETURN
6000 DATA 0000,0001,0010,0011
6010 DATA 0100,0101,0110,0111
6020 DATA 1000,1001,1010,1011
6030 DATA 1100,1101,1110,1111
6040 DATA 00,01,02,03,04,05,06,07
6050 DATA 08,09,0A,0B,0C,0D,0E,0F
6060 DATA 10,11,12,13,14,15,16,17
6070 DATA 18,19,1A,1B,1C,1D,1E,1F
6080 DATA 20,21,22,23,24,25,26,27
6090 DATA 28,29,2A,2B,2C,2D,2E,2F
6100 DATA 30,31,32,33,34,35,36,37
```

Listing 6-3—cont. Dec/Hex/Bin Conversion Program

```
6110 DATA 38,39,3A,3B,3C,3D,3E,3F
6120 DATA 40,41,42,43,44,45,46,47
6130 DATA 48,49,4A,4B,4C,4D,4E,4F
6140 DATA 50,51,52,53,54,55,56,57
6150 DATA 58,59,5A,5B,5C,5D,5E,5F
6160 DATA 60,61,62,63,64,65,66,67
6170 DATA 68,69,6A,6B,6C,6D,6E,6F
6180 DATA 70,71,72,73,74,75,76,77
6190 DATA 78,79,7A,7B,7C,7D,7E,7F
6200 DATA 80,81,82,83,84,85,86,87
6210 DATA 88,89,8A,8B,8C,8D,8E,8F
6220 DATA 90,91,92,93,94,95,96,97
6230 DATA 98,99,9A,9B,9C,9D,9E,9F
6240 DATA A0,A1,A2,A3,A4,A5,A6,A7
6250 DATA A8,A9,AA,AB,AC,AD,AE,AF
6260 DATA B0,B1,B2,B3,B4,B5,B6,B7
6270 DATA B8,B9,BA,BB,BC,BD,BE,BF
6280 DATA C0,C1,C2,C3,C4,C5,C6,C7
6290 DATA C8,C9,CA,CB,CC,CD,CE,CF
6300 DATA D0,D1,D2,D3,D4,D5,D6,D7
6310 DATA D8,D9,DA,DB,DC,DD,DE,DF
6320 DATA E0,E1,E2,E3,E4,E5,E6,E7
6330 DATA E8,E9,EA,EB,EC,ED,EE,EF
6340 DATA F0,F1,F2,F3,F4,F5,F6,F7
6350 DATA F8,F9,FA,FB,FC,FD,FE,FF
```

OUTLINER PROGRAM

What it does:

The Outliner program was designed to help you organize ideas. You enter one or more lines of text information which is stored with a unique "tag." You can enter up to 25 sets of tags and data. Outliner lets you:

1. Enter a new outline.
2. Load an existing outline from tape/disk.
3. Save the current outline to tape/disk.
4. Edit the current outline, adding more tags and data, erasing tags and data, or rearranging the order of the tags and data.
5. Display the tags at the screen.
6. Display the outline at the screen.
7. Print the tags at a printer.
8. Print the outline at a printer.

You can use any unique identifiers for the tags. After you enter a tag, enter as much text as you want to associate with the particular tag. You can enter one line of text for a tag or several lines. It depends on what type of information you're organizing.

Once you've entered your information, you can easily add more data, erase currently included data, or rearrange the information.

You rearrange information by selecting Edit Outline and entering two tags. The tags are exchanged but the information associated with each tag stays with the original tag. If you want to rearrange several sets of information, you must rearrange the data in groups of two tags.

Suppose you have two tags FIRST and FOURTH. FIRST has three items (1 to 3) and FOURTH has five items (19 to 23). When you rearrange these two tags, FOURTH and FIRST are exchanged but the tag's information is not changed. The information associated with FIRST is still associated with FIRST. This means that when you print/display your tags, FOURTH will list before FIRST.

When you erase a tag, all the information associated with the tag is also erased. When you add tags, you add as much information as you want for each new tag.

What it shows you:

The Outliner program shows you a way to process related information by using the concept of *pointers*. A pointer tells you where the information is located—a pointer "points" to other data.

The actual information data for the outline is stored in the DATA$ array. The elements of this array are not changed except when you erase a tag and its information.

The tags are all stored in the TAG$ array. This array is rearranged when you rearrange the outline. When you change the order of the tags, the tag identifiers are actually moved in the TAG$ array. This means that you can print the TAG$ array and see the actual order of the tags. The index (array number) for a tag is the same as the first index of the SEQ array that contains the pointers to the DATA$ array.

The SEQ array is actually a set of pointers. There are two dimensions to this array, SEQ(25,2). The first dimension matches the dimension of the TAG$ array. There's a direct connection between these arrays, as previously stated.

There are two elements in the SEQ array for each tag. The first element, SEQ(n,1), is used for the pointer data—it points to the first element in DATA$ associated with TAG$(n). The second element, SEQ(n,2), tells you how many elements of DATA$ go with TAG$(n).

Suppose you're looking at tag No. 5 which you called FIVE. There are ten entries for tag FIVE. The information starts at DATA$(28) and ends at DATA$(37).

TAG$(5) contains the actual tag identifier, "FIVE". SEQ(5,1) contains a pointer to the first data in the DATA$ array for the tag—28. SEQ(5,2) contains the number of elements in DATA$, beginning with the one pointed to by SEQ(5,1), that go with the tag—10.

You can see how easy it is to find the information for a specific tag. You look for the tag in TAG$ and use the TAG$ array element number as the index into the SEQ array. Then, SEQ(n,1) tells you where in the DATA$ array you will find the beginning of the tag's data and SEQ(n,2) tells you how many DATA$ elements you have to process.

You can easily reorder the information without having to move the actual string data in DATA$. You just change the pointers in the SEQ array.

What you can do:

If you have Extended BASIC and more memory, you can easily increase the values for the DATA$, TAG$, and SEQ arrays. If you don't have more memory, you have to analyze the way you're using the program before deciding which arrays to increase.

Depending on the way you store information with the tags, you may want to increase the number of elements in the DATA$ array. You can now store a maximum of 100 elements, or approximately 4 per tag.

Table 6-7. Outliner Routines

Lines	Function
100-350	Initialization and print menu.
1000-1050	Process new outline.
2000-2290	Load an existing outline file.
3000-3290	Save the current outline to a file.
4000-4110	Edit the current outline.
5000-5050	Display the tags at the screen.
6000-6050	Display the outline at the screen.
7000-7060	Print the tags.
8000-8060	Print the outline.
9000-9200	Get tag and outline data.
10000-10060	Change the outline title.
11000-11020	Add tags and data to the current outline.
12000-12140	Erase tags and data.
13000-13250	Rearrange tags and data.
14000-14080	Edit tags and data.
15000-15140	Display/print tags.
16000-16280	Display/print outline data.
17000-17080	Search for a tag.
18000-18050	Dump tag and data for a particular tag.
19000-19160	Erase a tag and data.
20000-20110	Change a tag and data.
21000-21040	Delay routine

If you find that you're storing many small lines with each tag, you can increase the DATA$ array. Remember to increase the value for MAXDTA to the same value as the bound (maximum subscript value) of the DATA$ array.

You may want to increase the number of tags that you can store. To do this, increase the TAG$ and SEQ$ arrays. The first bound of the SEQ array must agree with the bound of the TAG$ array. That means that if you increase TAG$ to TAG$(50), you must also increase SEQ to SEQ(50,2). Remember to increase MAXTAG to whatever the bound is for TAG$.

Table 6-8. Outliner Variables

Variable	Description
ANS	Used in getting information from a menu.
DATA$(100)	Character data associated with the tags.
DEV	Device value used in printing the outline and tags. Device=0 for the screen. Device=1 for the printer.
FILE$	Filename for the saved/restored outline data.
FND	Used to search for a tag. FND is the array element number for the found tag.
INCR	Used to delete outline information.
MAXDTA	Maximum number of elements in the DATA$ array (currently 100).
MAXL	Maximum number of lines printed before a heading line is printed. (MAXL=20 for the screen, 55 for the printer.)
MAXTAG	Maximum number of tags processed (currently 25).
NUMDTA	Number of elements in the DATA$ array for the outline being processed.
NUMTAG	Number of elements in the TAG$ array for the outline being processed.
PRT	Number of lines printed at the screen or printer.
SEQ(25,2)	Sequence array showing the first element in the DATA$ array for a particular tag and the number of elements in DATA$ associated with that tag. For example, if TAG # 3 has 6 lines of data associated with it and the first line is stored in DATA$(24), the SEQ array would be: SEQ(3,1)=24 and SEQ(3,2)=6.
SRCH$	Used to search for a tag.
TAG$(25)	The "tags" associated with the data in the outline.
TMP	Used to rearrange outline data.
TMP$	Used to rearrange tag data.

Listing 6-4. Outliner Program

```
100 REM OUTLINER PROGRAM
110 OPTION BASE 1
120 DIM TAG$(25),SEQ(25,2),DATA$(100)
130 MAXTAG=25
140 MAXDTA=100
150 CALL CLEAR
160 PRINT "*** OUTLINER PROGRAM ***": : :
170 PRINT "This program helps you to":
    "arrange/re-arrange"
180 PRINT "information in an outline."
190 PRINT :"You enter a 3-character tag":
    "and as much data as you want"
200 PRINT "for each tag.  Then, you can":
    "print the information,"
210 PRINT "re-arrange it by re-ordering":
    "the tags, add or erase data."
220 PRINT : :
230 GOSUB 21000
240 CALL CLEAR
250 PRINT : :"Program options:": :
    " 1. Process new outline"
260 PRINT " 2. Load existing outline":
    " 3. Save current outline"
270 PRINT " 4. Edit outline":" 5. Display tags":
    " 6. Display outline"
280 PRINT " 7. Print tags":" 8. Print outline":
    " 9. STOP"
290 PRINT
300 INPUT "Your choice -> ":ANS
310 IF (ANS<1)+(ANS>9)THEN 250
320 IF ANS<9 THEN 340
330 STOP
340 ON ANS GOSUB 1000,2000,3000,4000,5000,6000,
    7000,8000
350 GOTO 250
1000 REM PROCESS NEW OUTLINE
1010 NUMTAG=0
1020 NUMDTA=0
1030 INPUT "Enter outline title -> ":TITLE$
1040 GOSUB 9000
1050 RETURN
2000 REM LOAD EXISTING OUTLINE
2010 PRINT :"Your outline can be on":" 1. Cassette":
     " 2. Disk"
2020 PRINT " 3. Other":" 4. None":
2030 INPUT "Your choice -> ":ANS
2040 IF (ANS<1)+(ANS>4)THEN 2000
2050 IF ANS<4 THEN 2070
2060 RETURN
2070 ON ANS GOTO 2080,2100,2140
2080 OPEN #10:"CS1",INPUT ,INTERNAL,FIXED 128
2090 GOTO 2170
2100 PRINT "Enter file name as":"DSKn.filename"
2110 INPUT "Your filename -> ":FILE$
2120 OPEN #10:FILE$,INPUT ,INTERNAL
2130 GOTO 2170
2140 PRINT "Enter file name as":"device.filename"
```

Listing 6-4—cont. Outliner Program

```
2150 INPUT "Your filename -> ":FILE$
2160 OPEN #10:FILE$,INPUT ,INTERNAL
2170 INPUT #10:NUMTAG,NUMDTA
2180 INPUT #10:TITLE$
2190 FOR I=1 TO NUMTAG
2200 INPUT #10:TAG$(I)
2210 NEXT I
2220 FOR I=1 TO NUMTAG
2230 INPUT #10:SEQ(I,1),SEQ(I,2)
2240 NEXT I
2250 FOR I=1 TO NUMDTA
2260 INPUT #10:DATA$(I)
2270 NEXT I
2280 CLOSE #10
2290 RETURN
3000 REM SAVE CURRENT OUTLINE
3010 PRINT :"Your outline can be on":" 1. Cassette":
     " 2. Disk"
3020 PRINT " 3. Other":" 4. None": :
3030 INPUT "Your choice -> ":ANS
3040 IF (ANS<1)+(ANS>4)THEN 3000
3050 IF ANS<4 THEN 3070
3060 RETURN
3070 ON ANS GOTO 3080,3100,3140
3080 OPEN #10:"CS1",OUTPUT,INTERNAL,FIXED 128
3090 GOTO 3170
3100 PRINT "Enter file name as":"DSKn.filename"
3110 INPUT "Your filename -> ":FILE$
3120 OPEN #10:FILE$,OUTPUT,INTERNAL
3130 GOTO 3170
3140 PRINT "Enter file name as":"device.filename"
3150 INPUT "Your filename -> ":FILE$
3160 OPEN #10:FILE$,OUTPUT,INTERNAL
3170 PRINT #10:NUMTAG,NUMDTA
3180 PRINT #10:TITLE$
3190 FOR I=1 TO NUMTAG
3200 PRINT #10:TAG$(I)
3210 NEXT I
3220 FOR I=1 TO NUMTAG
3230 PRINT #10:SEQ(I,1),SEQ(I,2)
3240 NEXT I
3250 FOR I=1 TO NUMDTA
3260 PRINT #10:DATA$(I)
3270 NEXT I
3280 CLOSE #10
3290 RETURN
4000 REM EDIT OUTLINE
4010 PRINT :"Processing outline ";TITLE$
4020 PRINT :"Options are:":" 1. Change title":
     " 2. Add tags/data"
4030 PRINT " 3. Erase tags/data":
     " 4. Re-arrange tags/data"
4040 PRINT " 5. Edit tags/data":" 6. Done"
4050 PRINT
4060 INPUT "Your choice -> ":ANS
4070 IF (ANS<1)+(ANS>6)THEN 4000
4080 IF ANS<6 THEN 4100
```

Listing 6-4—cont. Outliner Program

```
4090 RETURN
4100 ON ANS GOSUB 10000,11000,12000,13000,14000
4110 GOTO 4000
5000 REM DISPLAY CURRENT TAGS
5010 DEV=0
5020 MAXL=20
5030 GOSUB 15000
5040 GOSUB 21000
5050 RETURN
6000 REM DISPLAY CURRENT OUTLINE
6010 DEV=0
6020 MAXL=20
6030 GOSUB 16000
6040 GOSUB 21000
6050 RETURN
7000 REM PRINT CURRENT TAGS
7010 DEV=1
7020 MAXL=55
7030 OPEN #1:"RS232.BA=10600",OUTPUT
7040 GOSUB 15000
7050 CLOSE #1
7060 RETURN
8000 REM PRINT CURRENT OUTLINE
8010 DEV=1
8020 MAXL=55
8030 OPEN #1:"RS232.BA=9600",OUTPUT
8040 GOSUB 16000
8050 CLOSE #1
8060 RETURN
9000 REM GET OUTLINE TAGS AND DATA
9010 NUMTAG=NUMTAG+1
9020 IF NUMTAG<=MAXTAG THEN 9050
9030 PRINT "Too many tags."
9040 RETURN
9050 INPUT "Enter tag (<ENTER> to end) ->":TAG$(NUMTAG)
9060 IF LEN(TAG$(NUMTAG))>0 THEN 9090
9070 NUMTAG=NUMTAG-1
9080 RETURN
9090 SEQ(NUMTAG,1)=NUMDTA+1
9100 PRINT "Now enter your data":
     "Enter <ENTER> to mark":
9110 PRINT "the end of the data for":"tag ";
     TAG$(NUMTAG)
9120 NUMDTA=NUMDTA+1
9130 IF NUMDTA<=MAXDTA THEN 9160
9140 PRINT "Too much data."
9150 RETURN
9160 INPUT DATA$(NUMDTA)
9170 IF LEN(DATA$(NUMDTA))>0 THEN 9120
9180 SEQ(NUMTAG,2)=NUMDTA-SEQ(NUMTAG,1)
9190 NUMDTA=NUMDTA-1
9200 GOTO 9000
10000 REM CHANGE TITLE
10010 PRINT "Current title is: ";TAB(3);TITLE$
10020 INPUT "Change it (Y/N) -> ":Y$
10030 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")
      THEN 10050
```

Listing 6-4—cont. Outliner Program

```
10040 RETURN
10050 INPUT "Enter new title -> ":TITLE$
10060 RETURN
11000 REM ADD TAGS/DATA
11010 GOSUB 9000
11020 RETURN
12000 REM ERASE TAGS/DATA
12010 GOSUB 5000
12020 PRINT "All data is erased when":
      "you erase a tag.": :
12030 INPUT "Erase which tag (<ENTER> to end) -> ":
      SRCH$
12040 IF LEN(SRCH$)>0 THEN 12060
12050 RETURN
12060 GOSUB 17000
12070 IF FND=0 THEN 12010
12080 PRINT "Data for ";SRCH$
12090 GOSUB 18000
12100 INPUT "Erase this tag/data (Y/N) -> ":Y$
12110 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")
      THEN 12130
12120 RETURN
12130 GOSUB 19000
12140 RETURN
13000 REM RE-ARRANGE TAGS/DATA
13010 PRINT "To re-arrange tags, enter":
      "two tags.  The tags":
13020 PRINT "and data get exchanged.": :
13030 INPUT "Enter tag 1 -> ":SRCH$
13040 IF LEN(SRCH$)>0 THEN 13060
13050 RETURN
13060 GOSUB 17000
13070 IF FND=0 THEN 13030
13080 GOSUB 18000
13090 FND1=FND
13100 INPUT "Enter tag 2 -> ":SRCH$
13110 IF LEN(SRCH$)>0 THEN 13130
13120 RETURN
13130 GOSUB 17000
13140 IF FND=0 THEN 13100
13150 GOSUB 18000
13160 TMP1=SEQ(FND1,1)
13170 TMP2=SEQ(FND1,2)
13180 SEQ(FND1,1)=SEQ(FND,1)
13190 SEQ(FND1,2)=SEQ(FND,2)
13200 SEQ(FND,1)=TMP1
13210 SEQ(FND,2)=TMP2
13220 TMP$=TAG$(FND)
13230 TAG$(FND)=TAG$(FND1)
13240 TAG$(FND1)=TMP$
13250 RETURN
14000 REM EDIT TAGS/DATA
14010 PRINT "Edit data for which tag"
14020 INPUT "<ENTER> to end -> ":SRCH$
14030 IF LEN(SRCH$)>0 THEN 14050
14040 RETURN
14050 GOSUB 17000
```

Listing 6-4—cont. Outliner Program

```
14060 IF FND=0 THEN 14000
14070 GOSUB 20000
14080 RETURN
15000 REM DISPLAY/PRINT TAGS
15010 PRT=60
15020 FOR I=1 TO NUMTAG
15030 IF PRT<MAXL THEN 15050
15040 GOSUB 15090
15050 PRINT #DEV:I;TAB(4);TAG$(I)
15060 PRT=PRT+1
15070 NEXT I
15080 RETURN
15090 REM PRINT HEADINGS
15100 IF DEV=1 THEN 15120
15110 CALL CLEAR
15120 PRINT #DEV:CHR$(140);"SEQ  TAG ID"
15130 PRT=1
15140 RETURN
16000 REM DISPLAY/PRINT OUTLINE
16010 PRT=0
16020 IF DEV=0 THEN 16040
16030 GOSUB 16190
16040 FOR I=1 TO NUMTAG
16050 IF PRT<MAXL THEN 16070
16060 GOSUB 16190
16070 PRINT #DEV
16080 PRINT #DEV:I;TAB(5);TAG$(I)
16090 PRT=PRT+2
16100 FOR J=SEQ(I,1)TO SEQ(I,1)+SEQ(I,2)-1
16110 N=INT(LEN(DATA$(J))/28)+1
16120 IF (PRT+N)<MAXL THEN 16140
16130 GOSUB 16190
16140 PRINT #DEV:DATA$(J)
16150 PRT=PRT+N
16160 NEXT J
16170 NEXT I
16180 RETURN
16190 REM PRINT HEADINGS
16200 IF DEV=0 THEN 16250
16210 PRINT #DEV:CHR$(140);TITLE$
16220 PRINT
16230 PRT=2
16240 RETURN
16250 GOSUB 21000
16260 PRT=2
16270 PRINT
16280 RETURN
17000 REM SEARCH FOR A TAG
17010 FND=0
17020 FOR I=1 TO NUMTAG
17030 IF SRCH$<>TAG$(I)THEN 17060
17040 FND=I
17050 RETURN
17060 NEXT I
17070 PRINT "Tag ";SRCH$;" not found."
17080 RETURN
18000 REM DUMP TAG AND DATA
```

Listing 6-4—cont. Outliner Program

```
18010 PRINT TAG$(FND)
18020 FOR J=SEQ(FND,1)TO SEQ(FND,1)+SEQ(FND,2)-1
18030 PRINT #DEV:DATA$(J)
18040 NEXT J
18050 RETURN
19000 REM ERASE A TAG AND DATA
19010 FOR I=1 TO NUMTAG
19020 IF SEQ(I,1)<=SEQ(FND,1)THEN 19040
19030 SEQ(I,1)=SEQ(I,1)-SEQ(FND,2)
19040 NEXT I
19050 INCR=SEQ(FND,2)
19060 FOR I=SEQ(FND,1)TO NUMDTA-INCR
19070 DATA$(I)=DATA$(I+INCR)
19080 NEXT I
19090 NUMDTA=NUMDTA-INCR
19100 FOR I=FND+1 TO NUMTAG
19110 SEQ(I-1,1)=SEQ(I,1)
19120 SEQ(I-1,2)=SEQ(I,2)
19130 TAG$(I-1)=TAG$(I)
19140 NEXT I
19150 NUMTAG=NUMTAG-1
19160 RETURN
20000 REM EDIT TAG/DATA
20010 PRINT "Tag is ";TAG$(FND)
20020 INPUT "Change it (Y/N) ->":Y$
20030 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")
      THEN 20040 ELSE 20050
20040 INPUT "Enter new tag -> ":TAG$(FND)
20050 FOR I=SEQ(FND,1)TO SEQ(FND,1)+SEQ(FND,2)-1
20060 PRINT "Data is :";DATA$(I)
20070 INPUT "Change it (Y/N) ->":Y$
20080 IF (SEG$(Y$,1,1)="Y")+(SEG$(Y$,1,1)="y")
      THEN 20090 ELSE 20100
20090 INPUT "Enter new data ->":DATA$(I)
20100 NEXT I
20110 RETURN
21000 REM DELAY ROUTINE
21010 PRINT "Press any key to continue.";
21020 CALL KEY(0,K,S)
21030 IF S=0 THEN 21020
21040 RETURN
```

Appendix

BASIC Statement, Command, and Function Formats

Notation:

words in **BOLDFACE AND CAPITALS** are *keywords* that you enter exactly as they appear.
num-exp means any *numeric expression,* like A+B, 42.34
num-var means any *numeric variable,* like X, INTEREST
str-exp means any *string expression,* like A$, "XYZ", FIRST$&MIDDLE$&LAST$
str-var means any *string variable,* like Y$, NAME$
variable means *any* variable, string or numeric, like YES$, PAYMENT
brackets ([ ]) means whatever is between the [ ] is *optional* and you don't have to use it if you don't want to
ellipsis (,. . .) means that the preceding thing can be repeated as many times as you want
device-filename means the device for cassette files (like CS1) and, for disk files, the name of the file on the disk as well as the device name (like DSK1.MYFILE)

ABS(*num-exp*)

Type: **Function**
Description: **ABS** returns the absolute value (positive value) of *num-exp.*

* * * * * * *

ASC(*str-exp*).

Type: **Function**
Description: **ASC** returns the ASCII value of the first character of *str-exp.*

* * * * * * *

ATN(*num-exp*)

Type: **Function**
Description: **ATN** returns the arctangent of *num-exp*. An arctangent is a trigonometric function.

* * * * * * *

BREAK [*line-num-list*]

Type: **Command or Statement**
Description: **BREAK** makes the program stop until you enter a **CONTINUE** command. If you use **BREAK** with a list of line numbers *(line-num-list)*, the program will stop when it reaches any of the lines included in *line-num-list*. **UNBREAK** makes all **BREAK** commands inactive.

* * * * * * *

BYE

Type: **Command**
Description: **BYE** closes all open files and leaves BASIC.

* * * * * * *

CALL CHAR(*ASCII-code,pattern-string*)

Type: **Command or Statement**
Description: **CHAR** redefines the pattern or image associated with the character represented by *ASCII-code*. The new pattern is given in *pattern-string*.

* * * * * * *

CHR$(*num-exp*)

Type: **Function**
Description: **CHR$** gives a one character string whose ASCII value is *num-exp*.

* * * * * * *

CALL CLEAR

Type: **Statement or Command**
Description: **CLEAR** "clears the screen" to all blank characters.

* * * * * * *

CLOSE # *file-number* **[:DELETE]**

Type: **Statement or Command**
Description: **CLOSE** closes file *file-number* and, if you say **DELETE,** removes the file from the device. You can only delete files from a disk. If you say **DELETE** with a cassette file, the file is closed but not removed from the tape.

* * * * * * *

CALL COLOR(*char-set,foreground-color,background-color***)**

Type: **Statement or Command**
Description: **COLOR** sets the foreground and background colors for the characters in *char-set.*

* * * * * * *

CONTINUE
or
CON

Type: **Command**
Description: The **CONTINUE** or **CON** command resumes executing a program after the program has executed a **BREAK** statement/command, had an error occur, or you press **FCTN CLEAR.** You can't CONTINUE a program after you've edited it.

* * * * * * *

COS(*num-exp***)**

Type: **Function**
Description: **COS** returns the trigonometric cosine of the value *num-exp* where *num-exp* is expressed in radians.

* * * * * * *

DATA *data-list*

Type: **Statement**
Description: A **DATA** statement stores numeric and/or string data in a program. You use a **READ** statement to put the values in the **DATA** statement *data-list* into variables in your program.

* * * * * * *

DEF *fctn-name*[*(parameter)*] = *expression*

Type: **Statement**
Description: **DEF** defines a numeric or string function with the name *fctn-name*. You can pass a value to the function with *parameter*. The value returned by the function is defined by *expression*. You use a function instead of rewriting *expression* every place you need it. This saves space in your program and makes it easy to change *expression*.

* * * * * * *

DELETE *"device-filename"*

Type: **Command or Statement**
Description: **DELETE** deletes file *filename* from *device*. You cannot use DELETE with a cassette.

* * * * * * *

DIM *array-name(bound1*[*,bound2,. . .*]) [*. . .*]

Type: **Statement or Command**
Description: **DIM** allocates (dimensions) space for the arrays. Each array *(array-name)* gets the number of elements *(bound1. . .)* allocated. You can have up to 3 dimensions in TI BASIC.

* * * * * * *

DISPLAY [*list*]

Type: **Statement or Command**
Description: **DISPLAY** writes data in *list* to the screen like a **PRINT** statement.

* * * * * * *

EDIT[*line-num*]

Type: **Command**
Description: **EDIT** allows you to edit line *line-num*.

* * * * * * *

END

Type: **Command or Statement**
Description: **END** ends your program and stops its execution. You can only use one **END** statement per program, as the last line in your program.

* * * * * * *

EOF [*(file-num)*]

Type: **Function**
Description: The **EOF** function tells you if you're at the end of the file *file-num*. If you're not, you get a 0. If you are, you get a 1. If there's no more room on the disk, you get a −1. The EOF function does not work with cassette files.

* * * * * * *

EXP(*num-exp*)

Type: **Function**
Description: **EXP** returns the exponential value of *num-exp*. This is e^x where e=2.718281828.

* * * * * * *

FOR *control* = *init-val* **TO** *end-val* [**STEP** *incr*]

Type: **Statement**
Description: **FOR-TO-STEP** repeatedly executes the statements between the **FOR** and **NEXT** statements. The control variable *control* starts at *init-val*. Each time the associated **NEXT** statement is executed, *control* is incremented by *incr* or by one (if you don't use **STEP**). If *control* is less than *end-val*, the statement between **FOR** and **NEXT** are executed again.

* * * * * * *

CALL GCHAR *(row,col,num-var)*

Type: **Statement or Command**
Description: **GCHAR** puts the ASCII code for the character at position *row* and *col* into *num-var.*

* * * * * * *

GOSUB *line-num*
or
GO SUB *line-num*

Type: **Statement**
Description: **GOSUB** transfers control to the subprogram at line *line-num.*

* * * * * * *

GOTO *line-num*
or
GO TO *line-num*

Type: **Statement**
Description: **GOTO** unconditionally transfers control to the statement at line *line-num*

* * * * * * *

CALL HCHAR *(row,col,ASCII-code[,repetitions])*

Type: **Statement or Command**
Description: **HCHAR** writes the character with value *ASCII-code* at row *row* and column *col.* If you use a value for *repetitions,* you'll get that many characters written across the screen beginning at *row,col.*

* * * * * * *

IF *condition* **THEN** *line-num1* [**ELSE** *line-num2*]

Type: **Statement**
Description: **IF-THEN-ELSE** determines if *condition* is true or false and transfers control to line number *line-numl* when the condition is true or line number *line-num2* when the condition is false.

* * * * * * *

INPUT [*prompt:*] *variable-list*

Type: **Statement**
Description: **INPUT** writes a message *(prompt)* to the screen and reads data into the variables in *variable-list.*

* * * * * * *

INPUT# *file-num* : *variable-list*

Type: **Statement**
Description: **INPUT#** reads data from file *file-num* into the variables in *variable-list.*

* * * * * * *

INT(*num-exp*)

Type: **Function**
Description: **INT** returns the integer value that is the largest integer less than or equal to *num-exp.*

* * * * * * *

CALL JOYST(*key-unit,x-return,y-return*)

Type: **Statement or Command**
Description: **JOYST** tells you the position of either joystick *(key-unit* = 1 or 2).

* * * * * * *

CALL KEY (*key-unit,return-var,status-var*)

Type: **Statement or Command**
Description: **KEY** returns the ASCII value of the key pressed in the *key-unit.*

* * * * * * *

LEN (*str-exp*)

Type: **Function**
Description: **LEN** returns the number of characters in *str-exp.*

* * * * * * *

[**LET**] *variable* = *expression*

Type: **Statement or Command**
Description: **LET** assigns the value of *expression* to *variable*. The keyword **LET** is an optional part of an assignment statement.

* * * * * * *

LIST [[*start-line*] [– [*end-line*]]]

Type: **Command**
Description: **LIST** lists the lines from the BASIC program in memory, beginning with line *start-line* and ending with line *end-line*. If you don't use *start-line*, the first line in the program is used. If you don't use *end-line*, the last line in the program is used.

* * * * * * *

LOG(*num-exp*)

Type: **Function**
Description: **LOG** returns the natural logarithm of the expression *num-exp*.

* * * * * * *

NEW

Type: **Command**
Description: **NEW** clears the computer's memory, clears the screen, and gets ready to accept a new program.

* * * * * * *

NEXT [*control*]

Type: **Statement**
Description: **NEXT** goes with a **FOR-TO-STEP** statement and increments the *control* value in a **FOR** statement.

* * * * * * *

NUMBER [*start-line*[,*increment*]]
or
NUM [*start-line*[,*increment*]]

Type: **Command**
Description: **NUMBER /romor NUM** generates sequenced line numbers for entering a BASIC program. *start-line* is 100 if you don't say otherwise. *increment* is 10 if you don't specify a value.

* * * * * * *

OLD *device*[.*program-name*]

Type: **Command**
Description: **OLD** loads the BASIC program from device *device*. If you're loading a program from a disk, you need to use *program-name*.

* * * * * * *

ON *num-exp* **GOSUB** *line-num* ; [. . .]

Type: **Statement**
Description: **ON-GOSUB** transfers control to the subprogram at the line number corresponding to *num-exp*.

* * * * * * *

ON *num-exp* **GOTO** *line-num*[. . .]

Type: **Statement**
Description: **ON-GOTO** unconditionally transfers control to the line number in the position corresponding to *num-exp*.

* * * * * * *

OPEN *#file-num:device-filename*[,*file-org*][,*file-type*] [,*open-mode*] [,*record-type*]

Type: **Statement or Command**
Description: **OPEN** associates the specified file with number *file-num* and enables the program to read/write data from/to the file.

* * * * * * *

OPTION BASE 0
or
OPTION BASE 1

Type: **Statement**
Description: **OPTION BASE** sets the lowest subscript for all arrays to zero or one.

* * * * * * *

POS *(string1,string2,num-exp)*

Type: **Function**
Description: **POS** returns the position of the first occurrence of *string2* in *string1* starting at character *num-exp* in *string1*.

* * * * * * *

PRINT [*#file-num* [,**REC** *rec-num*] *:*] [*list*]

Type: **Statement or Command**
Description: **PRINT** writes the data in *list* to the screen or to the file *file-num*.

* * * * * * *

RANDOMIZE [*num-exp*]

Type: **Statement or Command**
Description: **RANDOMIZE** resets the random number generator.

* * * * * * *

READ *variable-list*

Type: **Statement**
Description: **READ** assigns values from a **DATA** statement to the variables in *variable-list*. (See the **DATA** statement.)

* * * * * * *

REM *string*

Type: **Statement or Command**
Description: **REM** lets you include remarks (nonexecutable statements) in a BASIC program. You use REMarks to tell what

your program is doing, how it operates, and what your variables are.

* * * * * * *

RESEQUENCE [*initial*] [,*increment*]
or
RES [*initial*] [,*increment*]

Type: **Command**
Description: **RES** or **RESEQUENCE** renumbers the lines in the BASIC program currently in memory. Line numbers start at *initial* (or 100) and increase by *increment* (or 10).

* * * * * * *

RESTORE [*line-num*]
or
RESTORE #*file-num* [,**REC** *rec-num*]

Type: **Statement or Command**
Description: **RESTORE** resets the line number for **DATA** statement used in the next **READ** statement. **RESTORE#** resets the current record number for file *file-num*.

* * * * * * *

RETURN

Type: **Statement**
Description: **RETURN** transfers program control from a subprogram to the statement following the **GOSUB** or **ON GOSUB** statement that called the subprogram.

* * * * * * *

RND

Type: **Function**
Description: **RND** returns a random number between 0 and 1.

* * * * * * *

RUN [*line-num*]

Type: **Command**
Description: **RUN** loads and executes the program in *device-filename*

or begins executing the BASIC program currently in memory, starting at *line-num*. If you just use **RUN**, the program in memory begins executing at its first line.

* * * * * * *

SAVE *device-filename*

Type: **Command**
Description: **SAVE** writes the BASIC program currently in memory to *device-filename*.

* * * * * * *

CALL SCREEN *(color-code)*

Type: **Statement or Command**
Description: **SCREEN** changes the screen color to that given by *color-code*.

* * * * * * *

SEG$ *(str-exp,position,length)*

Type: **Function**
Description: **SEG$** returns a substring of *str-exp*. The returned string is *length* characters long and begins at character *position* in *str-exp*.

* * * * * * *

SGN *(num-exp)*

Type: **Function**
Description: **SGN** returns a one if *num-exp* is positive, a zero if *num-exp* is zero, and a minus one if *num-exp* is negative.

* * * * * * *

SIN *(num-exp)*

Type: **Function**
Description: **SIN** returns the sine of *num-exp* where *num-exp* is in radians.

* * * * * * *

CALL SOUND *(duration,freq1,vol1*[*,. . . ,freq4,vol4*]*)*

Type: **Statement or Command**
Description: **SOUND** controls the tone and noise generator. You get a tone of frequency *freq1* at *vol1* for *duration* milliseconds. You can get up to four simultaneous tones (all for the same *duration*).

* * * * * * *

SQR *(num-exp)*

Type: **Function**
Description: **SQR** returns the square root of *num-exp*.

* * * * * * *

STOP

Type: **Statement or Command**
Description: **STOP** terminates program execution. You can use **STOP** statements anywhere in your program.

* * * * * * *

STR$ *(num-exp)*

Type: **Function**
Description: **STR$** converts *num-exp* into its string form. **VAL** works the other way, changing a string to a numeric form.

* * * * * * *

TAB *(num-exp)*

Type: **Function**
Description: **TAB** positions **PRINT** or **DISPLAY** statements at column *num-exp*.

* * * * * * *

TAN *(num-exp)*

Type: **Function**
Description: **TAN** returns the tangent of *num-exp* where *num-exp* is expressed in radians.

* * * * * * *

TRACE

Type: **Command or Statement**
Description: **TRACE** lists line numbers of statements before execution.

* * * * * * *

UNBREAK [*line-num-list*]

Type: **Command or Statement**
Description: **UNBREAK** removes the breakpoints for the lines in *line-num-list* or all breakpoints (if you don't use *line-num-list*). You set breakpoints with a **BREAK** command.

* * * * * * *

UNTRACE

Type: **Command or Statement**
Description: **UNTRACE** cancels a **TRACE** command. Line numbers are no longer printed before the statements are executed.

* * * * * * *

VAL *(str-exp)*

Type: **Function**
Description: **VAL** converts *str-exp* to a numeric form.

* * * * * * *

CALL VCHAR *(row,col,ASCII-code[,repetitions])*

Type: **Statement or Command**
Description: **VCHAR** writes the character with the ASCII value *ASCII-code* at row *row* and column *col*. If you use a value for *repetitions*, you'll get that many characters written down the screen beginning at *row,col*.

* * * * * * *

SAMS *PRESENTS*

More Books for TI 99/4A® Owners!

THE TI 99/4A USER'S GUIDE

You'll like this badly needed guide to all aspects of the TI 99/4A! Covers everything from system setup to expansion options, including languages, software, and peripherals. By Carol Ann Casciato and Donald J. Horsfall. 224 pages, 5½ × 8½, soft. ISBN 0-672-22071-7. © 1983.

Ask for No. 22071 ..... **$11.95**

TI 99/4A: 24 BASIC PROGRAMS

An inexpensive source of fun-and-useful, completely tested BASIC programs that take full advantage of your TI 99/4A's sound and graphics capabilities. Also covers fundamental programming commands, debugging, utilities, and more. By Carol Ann Casciato and Donald J. Horsfall.

BOOK ONLY: 160 pages, 5½ × 8½, comb. ISBN 0-672-22247-7. © 1983.

Ask for 22247 ..... **$12.95**

TAPE CASSETTE OF PROGRAMS ONLY: Saves you from typing-in each program listing. ISBN 0-672-22291-4.

Ask for No. 22291 ..... **$7.95**

BOOK PLUS TAPE CASSETTE: Packed in 6 × 9 hardcover vinyl binder with cassette storage feature. ISBN 0-672-26172-3.

Ask for No. 26172 ..... **$16.95**

TI 99/4A: 51 FUN AND EDUCATIONAL PROGRAMS

Run all 51 BASIC programs as-is on the TI 99/4A or adapt to almost any other computer. Begin with easy, short programs and progress to long, more complex ones. Ideal for first-time computer users of any age. By Gil M. Schechter.

BOOK ONLY: 80 pages, 5½ × 8½, soft. ISBN 0-672-22192-6. © 1983.

Ask for 22192 ..... **$4.95**

TAPE CASSETTE OF PROGRAMS ONLY: Saves you from typing-in each program listing. ISBN 0-672-22283-3.

Ask for No. 22283 ..... **$7.95**

BOOK PLUS TAPE CASSETTE: Packed in 6 × 9 hardcover vinyl binder with cassette storage feature. ISBN 0-672-26168-5.

Ask for No. 26168 ..... **$11.95**

ENTERTAINMENT GAMES IN TI BASIC AND EXTENDED BASIC

Highly organized, fully listed collection of 20 original game programs for the TI 99/4A computer, 9 of which are in standard TI BASIC with the remainder in Extended BASIC. Each line of code clearly explained. About half are arcade-type games, with the remainder assorted. By Khoa Ton and Quyen Ton.

BOOK ONLY: 176 pages, 5½ × 8½, soft. ISBN 0-672-22204-3. © 1983.
Ask for No. 22204 ..... **$8.95**

TAPE CASSETTE OF PROGRAMS ONLY: Saves you from typing-in each program listing. ISBN 0-672-22285-X.
Ask for No. 22285 ..... **$7.95**

BOOK PLUS TAPE CASSETTE: Packed in 6 × 9 hardcover vinyl binder with cassette storage feature. ISBN 0-672-26169-3.
Ask for No. 26169 ..... **$15.95**

USER'S GUIDE TO MICROCOMPUTER BUZZWORDS

Handy quick-reference that provides an understanding of the basic terminology you need to become "computer literate." Contains many illustrations. By David H. Dasenbrock. 110 pages, 5½ × 8½, soft. ISBN 0-672-22049-0. © 1983.
Ask for No. 22049 ..... **$9.95**

USING COMPUTER INFORMATION SERVICES

Shows you how to use your microcomputer to communicate with everything from local bulletin boards to the national computer networks. Clearly explains what to expect on-screen, how to retrieve it, and more. By Larry Sturtz and Jeff Williams. 240 pages, 5½ × 8½, soft. ISBN 0-672-21997-2. © 1983.
Ask for No. 21997 ..... **$12.95**

MEGABUCKS FROM YOUR MICROCOMPUTER

Shows you how to make money using your creative talents through your microcomputer, and includes details for doing your own writing, reviewing, and programming. By Tim Knight. 80 pages, 8½ × 11, soft. ISBN 0-672-22083-0. © 1983.
Ask for No. 22083 ..... **$3.95**

These and other Sams Books and Software products are available from better retailers worldwide, or directly from Sams. Call 800-428-SAMS or 317-298-5566 to order, or to get the name of a Sams retailer near you. Ask for your free Sams Books and Software Catalog!

Prices good in USA only. Prices and page counts subject to change without notice.

TI 99/4A is a registered trademark of Texas Instruments.

SAMS

TI-99/4A: 24 BASIC Programs

Containing 24 programs written in TI BASIC, this book covers a gamut from child/adult entertainment games to highly applicable household utility and service programs. Whether you categorize yourself as a novice or computer buff, with these programs and your TI-99/4A microcomputer, you can:

- **FIND** just the right entertaining game or helpful program
- **ENTER** any program into your TI-99/4A from the 24 listings provided
- **RUN** each of the programs for hours of fun and pleasure
- **PLAY** interesting games and programs on your TI-99/4A
- **LIST** each program line-by-line, analyze its logical construction
- **CHANGE** a program line to meet your particular need
- **LEARN** to program in TI BASIC, create custom programs
- **SAVE** your new or modified programs to cassette tapes for future use and enjoyment

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

$12.95/22247

ISBN: 0-672-22247-7