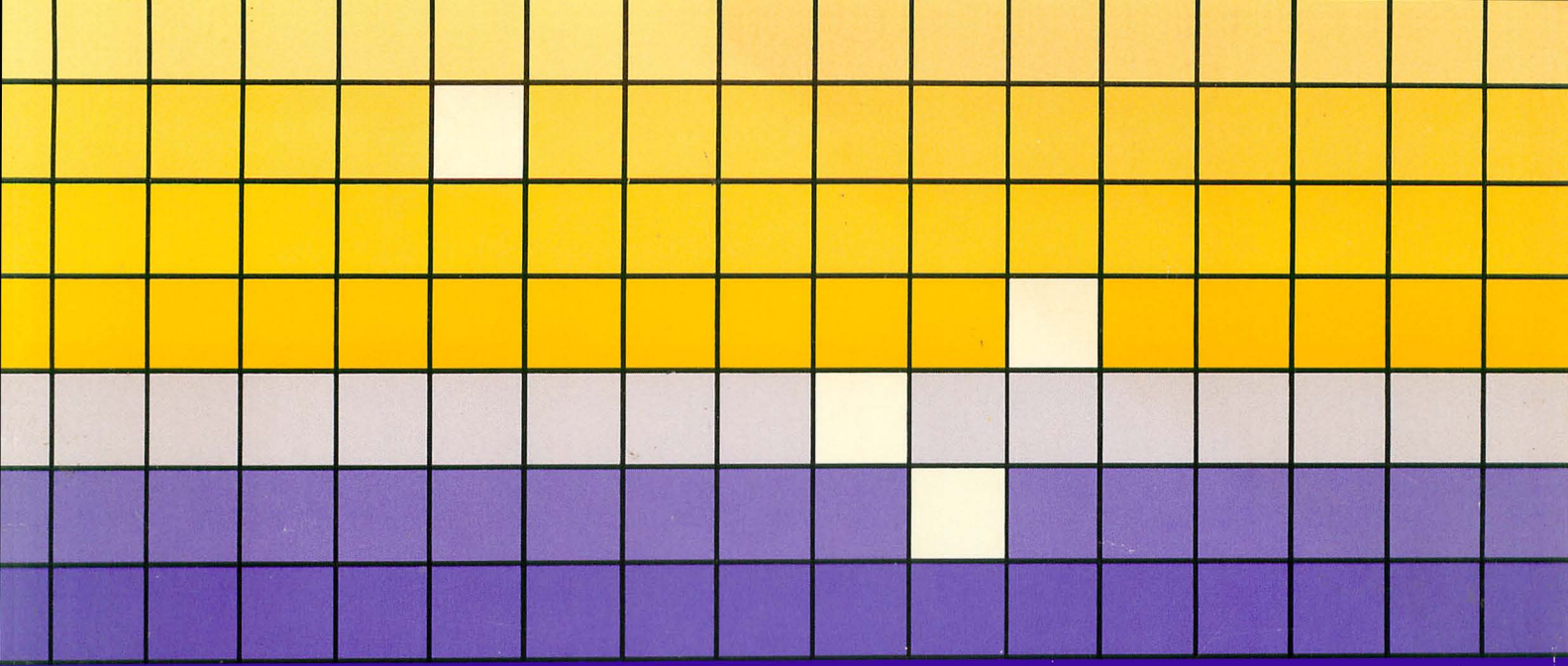

Home Applications and Games for the ATARI® Home Computers

For the ATARI® 400™/800™, 600XL™, 800XL™, 1200XL™, 1400XL™, and 1450XLD™ Home Computers

Little, Brown Microcomputer Bookshelf

Timothy P. Banse

K. Tom

Home Applications and Games for the ATARI® Home Computers

The Little, Brown Microcomputer Bookshelf

BANSE, TIMOTHY
Home Applications and Games for the VIC–20

BANSE, TIMOTHY
Home Applications and Games for the Apple® II Plus and Apple® IIe Computers

BANSE, TIMOTHY
Home Applications and Games for the ATARI® 400™/800™, 600XL™, 800XL™, 1200XL™, 1400XL™, and 1450XLD™ Home Computers

BARNETT, MICHAEL P. AND GRAHAM K. BARNETT
Personal Graphics for Profit and Pleasure on the APPLE® II Plus Computer

BARNETT, MICHAEL P. AND GRAHAM K. BARNETT
More Personal Graphics for Profit and Pleasure on the Apple® II Plus Computer

HODGES, WILLIAM AND NEAL NOVAK
Applications Software for Homes and Businesses

MORRILL, HARRIET
BASIC for the IBM Personal Computer

MORRILL, HARRIET
Mini and Micro BASIC: Introducing Applesoft®, Microsoft®, and BASIC PLUS

NAHIGIAN, J. VICTOR AND WILLIAM S. HODGES
Computer Games for Businesses, Schools, and Homes

NAHIGIAN, J. VICTOR AND WILLIAM S. HODGES
Computer Games for Business, School, and Home for TRS-80 Level II BASIC

ORWIG, GARY W. AND WILLIAM S. HODGES
The Computer Tutor: Learning Activities for Homes and Schools (for the TRS-80®, Apple®, and PET/CBM® Home Computers)

ORWIG, GARY W. AND WILLIAM S. HODGES
The Computer Tutor: ATARI® Home Computer Edition (for the ATARI® 400/800™, 600XL™, 800XL™, 1200XL™, 1400XL™, and 1450 XLD™ Home Computers and the ATARI® VCS 2600™/5200™ Computer Keyboards)

WINDEKNECHT, THOMAS G.
6502 Systems Programming

Home Applications and Games for the ATARI® Home Computers

FOR THE ATARI®
400™/800™, 600XL™, 800XL™,
1200XL™, 1400XL™, and 1450XLD™
HOME COMPUTERS

Timothy P. Banse

LITTLE, BROWN AND COMPANY
Boston Toronto

For my son, Christopher Malone-Banse

Library of Congress Cataloging in Publication Data

Banse, Timothy P.
Home applications and games for the Atari.

(The Little, Brown microcomputer bookshelf series)
1. Atari computer—Programming. 2. Computer programs.
3. Computer games. I. Title. II. Series.
QA76.8.A82B36 1983 001.64'2 83-18734
ISBN 0-316-08044-6

Library of Congress Catalog Card Number 83-18734

ISBN 0-316-08044-6

9 8 7 6 5 4 3 2 1

MV

Published simultaneously in Canada
by Little, Brown & Company (Canada) Limited

Printed in the United States of America

Chapters 6 and 17 first appeared, in a slightly different form, in *MicroComputing* magazine. Copyright © 1982, 1983 by *MicroComputing*. Used by permission.

How To Use This Book

This book provides a set of ready-to-run programs for use with your ATARI Home Computer. Each program comes with background information on the topic of the chapter, an explanation of how each program works, and a variable table that lists what each variable stands for. So it's possible either to jump into the book and simply ENTER and RUN a program, or to approach each chapter as a learning experience.

When you type in the programs, you'll notice that some are fairly long. Don't be discouraged. Leaving out the REM statements will shorten them considerably. It also makes the programs run faster, because there will be fewer lines for the microprocessor to read during execution of the program.

You'll also notice that some of the variable names, like SENTENCE$ in the Crypto-System program, are very long. The programs were written that way to make them easier to read and understand. You'll find it less tedious—and easier on available memory—to shorten variable names like SENTENCE$ to just the first few letters: SEN$.

When you enter the programs, use the ATARI BASIC abbreviations. Type RET. for RETURN, D. for DATA, and ? for PRINT. The abbreviations make a big difference in how long it takes to finish a program.

This book was written so that anyone owning an ATARI® 400™ Home Computer and a recorder can use it. What if you own an ATARI® 800™ Home Computer or any of the XL™ series Home Computers and a printer? No problem—all the programs will run fine. To use the printer, simply go into the body of the program and add: LPRINT WHATEVER$ you need printed.

If you'd rather forego the labor of typing in these programs, they're available from the author on disk or cassette for $9.95 (as of this writing) at: Version 1.0 Software, P.O. Box 5535, Coralville, Iowa 52241–0535.

Contents

1 / Checkbook Balancer

No doubt you've seen those fancy checkbook balancers designed to run on your micro. They keep the balance up to date, the number of each check, tell who was paid and on what date. The problem is, it takes longer to run the program than to simply figure it out in your head and write down the results with a ballpoint pen. Want to keep things simple? Here's a quick and easy way to keep the balance up to date. It's *fast* and *accurate*.

A checkbook balancer is a handy tool. With one, it takes very little work to keep the balance up to date, allowing you to always know exactly how much money you have in your bank account.

First, enter the current balance. Then, one at a time, enter the amount of each check you've written. If the balance drops below zero—forbidden by the bank—an overdrawn message will flash on the screen, and the border will turn red. The amount overdrawn will be displayed as the balance. If you need to find out the extent of the damages, keep entering the other checks and keep an eye on the total overdraft figure.

On the other hand, if the balance remains in the black, as each check is entered, it will be subtracted from the old balance, and the new amount will be tallied and printed on the screen. When the figuring is finished, pencil the new balance in the checkbook.

As it is constructed, this program will accommodate twenty-seven checkbook entries, printed in three columns. Beyond that, the display gets very messy. The remedy is to stop at the twenty-seventh entry, record the balance on paper, hit the BREAK key, then RUN with the latest balance.

Here's another use for the program. Use it as a budgeting tool to keep track of cash flow. At the end of the month, go through the

checkbook and enter just the checks written to a particular category, say, the grocery. When totaled, you'll know if you are exceeding the amount of dollars allocated for food, for entertainment, and so on.

For this application, instead of entering the true balance of the account at the start of the program, enter the amount budgeted for that category. If you go over the amount, the overdrawn message will flash on the screen, and the display will tell you how far over the limit you have wandered.

How the Program Works

Line 110 blanks the screen.

Line 130 dimensions the string variable BLANK$, giving it room for ten blank spaces.

Line 140 assigns BLANK$ as consisting of ten blank spaces.

Line 150 assigns the variable BALANCE the value of zero.

Line 160 assigns the variable COLUMN the value of zero.

Line 170 indents the text six spaces.

Line 190 prints CHECKBOOK BALANCER in INVERSE VIDEO. Press the INVERSE key once for INVERSE, once more to turn it off.

Line 220 asks for the current balance. Using a semicolon and a colon after a PRINT statement and before an INPUT prints the prompt (?) on the same line as the print statement.

Line 260 tells the program where to print BLANK$.

Line 270 prints BLANK$, erasing the last check entered.

Line 280 asks for the next check, at the same screen location as the last check entry.

Line 300 gets each check entry, one by one.

Line 320 numbers each check as it's entered.

Line 340 counts how many checks are printed in each column.

Line 370 checks to see if the column is full. If so, then starts printing in the next column.

Line 390, if a new column, starts the count over, so only nine checks are printed in each column.

Line 410 tells the ATARI Home Computer where to print the check number and its amount.

Line 440 prints the check number, followed by a period, a dollar sign, and the amount of the check.

Line 460 subtracts the latest check from the balance.

Line 480 positions the next BLANK$ so it will erase the old balance.

Line 490 prints BLANK$, erasing the old balance.

Line 500 positions the cursor to print the new balance.

Line 520 prints the new balance.

Line 540 checks the balance. If no money is left in the account, goes to the subroutine OVERDRAWN, at line 580.

Line 560 branches the program to line 260 for next entry.

Line 580 turns the screen border red.

Line 590 locates the cursor to print the next message.

Line 610 prints OVERDRAWN in inverse video.

Line 620 returns control to the main program loop.

String and Numeric Variables

OVERDRAWN	Line number of subroutine that draws pink border.
BLANK$	10 blank spaces used to erase text.
BALANCE	Tells how much money left in account.
COLUMN	Determines which of three columns to print check's amount.
CHECK	Dollar amount of last check written.
COUNT	How many checks printed in current column.
NUMBER	Number of checks figured so far.

The Program

```
100 REM CHECKBOOK BALANCER
110 PRINT CHR$(125):REM CLEAR SCREEN
120 OVERDRAWN=580:REM DEFINE SUBROUTINE LOCATION
130 DIM BLANK$(10)
140 BLANK$="          ":REM TEN BLANK SPACES
150 BALANCE=0
160 COLUMN=0
170 POKE 85,6
180 REM PRINT INVERSE VIDEO
```

```
190 PRINT "CHECKBOOK BALANCER"
200 REM GET CURRENT BALANCE
210 PRINT :PRINT
220 PRINT "CURRENT BALANCE  ";:INPUT BALANCE
230 PRINT :PRINT
240 PRINT "AMOUNT CHECK WRITTEN FOR"
250 REM ERASE ANY PREVIOUS ENTRY
260 POKE 84,8
270 PRINT BLANK$
280 POKE 84,8
290 REM GET NEXT CHECK
300 INPUT CHECK
310 REM CHECKS TALLIED IN SESSION
320 NUMBER=NUMBER+1
330 REM COUNT CHECKS PRINTED IN COLUMN
340 COUNT=COUNT+1
350 REM FORMAT PRINTING OF CHECKS WRITTEN
360 REM PRINT 9 CHECKS, THEN NEW COLUMN
370 IF COUNT=10 THEN COLUMN=COLUMN+13
380 REM START COUNT OVER
390 IF COUNT=10 THEN COUNT=1
400 REM PRINT IN COLUMNS
410 POSITION COLUMN,COUNT+12
420 REM PRINT NUMBER CHECKS CALCULATED
430 REM WITH AMOUNT WRITTEN FOR
440 PRINT NUMBER;". $";CHECK
450 REM FIGURE BALANCE
460 BALANCE=BALANCE-CHECK
470 REM ERASE ANY PREVIOUS ENTRY
480 POSITION 18,4
490 PRINT BLANK$
500 POSITION 18,4
510 REM PRINT NEW BALANCE
520 PRINT BALANCE
530 REM IF NO MONEY LEFT SHOW OVERDRAFT
540 IF BALANCE<0 THEN GOSUB OVERDRAWN
550 REM DO NEXT CHECK
560 GOTO 260
570 REM OVERDRAWN ROUTINE
580 SETCOLOR 4,4,4:REM MAKE SCREEN RED
590 POSITION 8,4
600 REM PRINT IN INVERSE VIDEO
610 PRINT "OVERDRAWN"
620 RETURN
```

2 / Budget Power

A budget is a simple concept. How much do the bills total during the budget period, and how much money is available to pay those liabilities? The budget power program lets you plug in those cash flow variables and make plans according to the results. Expenses are categorized in the way they are encountered by the average household, but it is easy to switch the categories you don't need for ones you do.

RUN the program and fill in how many dollars you have to cover the budget period. Fill in each budget item amount. Once these itemized expenses are totaled, a printout tells how much you need.

A budget can be manipulated as a creative tool. It need not be as rigid as the Federal government's. Figure yours a couple of different ways. Cut back on food and pay more on old loans with high interest rates. Consider the short- and long-term impact of your planning.

How the Program Works

Line 110 blanks the screen.

Line 130 locates the cursor at about mid-screen and one line down from the top.

Line 140 prints the program title: BUDGET.

Line 150 locates the cursor three lines down from the top of the screen and indented three spaces from the side.

Line 160 asks how much money is available for the budget period.

Line 180 asks how much is needed for shelter during that same budget period. Likewise, lines 200, 220, 240, 260, and so on, ask for the appropriate sums to handle each category of expense.

Lines 190, 230, 270, and the other lines using POSITION format the screen display. In the case of line 190, POSITION 20,5, the 20 indents the text twenty spaces from the left and the 5 locates the cursor five lines from the top of the screen.

Line 470 tallies each amount, and stores the result in the variable, TOTAL.

Line 480 moves the cursor to the top right of the screen.

Line 490 prints the TOTAL.

Line 510 locates the cursor at the bottom left of the screen.

String and Numeric Variables

MONEY	The amount available to finance the budget period.
SHELTER	Projected amount needed to cover this expense.
FOOD	Projected amount needed to cover this expense.
ELECTRIC	Projected amount needed to cover this expense.
PHONE	Projected amount needed to cover this expense.
CAR	Projected amount needed to cover this expense.
LOAN	Projected amount needed to cover this expense.
FUEL	Projected amount needed to cover this expense.
TIRES	Projected amount needed to cover this expense.
INSURANCE	Projected amount needed to cover this expense.

LIFE	Projected amount needed to cover this expense.
MED	Projected amount needed to cover this expense.
CLOTHES	Projected amount needed to cover this expense.
FUN	Projected amount needed to cover this expense.
GAMBLE	Projected amount needed to cover this expense.
SLUSH	Projected amount needed to cover this expense.
TOTAL	The sum of money needed, based on the total of each of the categories, to meet the budget period's deficits.

The Program

```
100 REM HOME BUDGET ANALYZER
110 PRINT CHR$(125):REM CLEAR SCREEN
120 POKE 82,0
130 POSITION 15,1
140 PRINT "BUDGET"
150 POSITION 3,3
160 PRINT "HAVE $";:INPUT MONEY
170 PRINT
180 PRINT "RENT/MORTGAGE";:INPUT SHELTER/
190 POSITION 20,5
200 PRINT "FOOD";:INPUT FOOD
210 PRINT
220 PRINT "UTILITIES";:INPUT ELECTRIC
230 POSITION 20,7
240 PRINT "TELEPHONE";:INPUT PHONE
250 PRINT
260 PRINT "LOAN (CAR)";:INPUT CAR
270 POSITION 20,9
280 PRINT "INSTALLMENT";:INPUT LOAN
290 PRINT
300 PRINT "GAS/DIESEL FUEL";:INPUT FUEL
310 POSITION 20,11
320 PRINT "MISC. (CAR)";:INPUT TIRES
330 PRINT
340 PRINT "INSURANCE (CAR)";:INPUT INSURANCE
350 POSITION 20,13
360 PRINT "LIFE";:INPUT LIFE
370 PRINT
380 PRINT "DENTAL/MEDICAL";:INPUT MED
390 POSITION 20,15
400 PRINT "KIDS CLOTHES";:INPUT CLOTHES
410 PRINT
```

```
420 PRINT "ENTERTAINMENT";:INPUT FUN
430 POSITION 20,17
440 PRINT "SAVINGS/INVEST.";:INPUT GAMBLE
450 PRINT
460 PRINT "MISC.  EXPENSE";:INPUT SLUSH
470 TOTAL=SHELTER+FOOD+ELECTRIC+PHONE+CAR+LOAN+FUEL+TIRES+INSURANCE+LIF
E+MED+CLOTHES+FUN+GAMBLE+SLUSH
480 POSITION 25,3
490 PRINT "NEED  $";TOTAL
500 REM GET CURSOR OUT OF WAY
510 POSITION 0,21
```

3 / Number Averaging

It often seems our world is made up of numbers. This can be good because numbers help us keep track of our favorite players' batting averages or yards rushing, as well as our grades at school.

This program makes it simple to figure the average for any kind of data. Just enter each number, one at a time. Each entry can be up to 97 places long. After the last of the numbers is entered, and you see the next question mark prompt, hit RETURN. The numbers will be tallied and displayed on the screen.

Here's what a sample run looks like.

```
? 1
? 2
? 47
? 10
? 5
? (carriage return)
TOTAL = 65
AVERAGE = 13
```

How the Program Works

Line 110 blanks the screen.

Line 130 dimensions the string variable NUMBER$. DIM (97) reserves space for a number up to 97 places long!

Line 180 inputs each number, one at a time, as a string variable.

Line 200 keeps track of how many numbers are entered.

Line 220 looks for a carriage return. If it finds one, the program quits taking numbers and averages the ones already entered.

Line 240 Remember we're inputting the numbers as string variables (NUMBER$). Since we perform mathematical functions on them, we convert string variables to numeric with NUMBER = VAL(NUMBER$). The string variable "47" becomes the number 47.

Line 260 adds each new number to the total.

Line 280 sends us back for each new number.

Line 300 is the beginning of the end. If line 220 found a carriage return, we subtract one from the number of entries, eliminating the carriage return as an entry.

Line 320 does the calculating.

Lines 340 through 370 print the sum of the numbers entered and their average.

String and Numeric Variables

NUMBER$	Used to hold individual numbers that will be averaged.
COUNT	Tells how many numbers have been entered for averaging.
NUMBER (NUMBER$)	Holds each number as a string variable. Once converted to a numeric, NUMBER holds its value.
TOTAL	Is the sum of all the numbers entered.
AVERAGE	The sum TOTAL divided by COUNT.

The Program

```
100 REM NUMBER AVERAGING DEVICE
110 PRINT CHR$(125):REM CLEAR SCREEN
120 REM ENTER ANY NUMBER UP TO 97 CHARS. LONG
130 DIM NUMBER$(97)
140 PRINT :PRINT
150 PRINT "WHAT NUMBERS"
160 PRINT
170 REM ENTER NUMBER
180 INPUT NUMBER$
190 REM KEEP TRACK OF HOW MANY ENTERED
```

```
200 COUNT=COUNT+1
210 REM HIT RETURN TO TALLY
220 IF NUMBER$="" THEN 290
230 REM CONVERT STRING VARIABLE TO NUMBER VALUE
240 NUMBER=VAL(NUMBER$)
250 REM ADD NEW NUMBER TO OLD TOTAL
260 TOTAL=TOTAL+NUMBER
270 REM GO BACK FOR NEXT NUMBER
280 GOTO 180
290 REM DECREMENT TO NULL CARRIAGE RETURN
300 COUNT=COUNT-1
310 REM DIVIDE TOTAL BY NUMBER ENTRIES
320 AVERAGE=TOTAL/COUNT
330 REM PRINT RESULTS OF AVERAGING
340 PRINT
350 PRINT "TOTAL   = ";TOTAL
360 PRINT
370 PRINT "AVERAGE = ";AVERAGE
```

4 / Calorie Counter

Want a trimmer waistline? The best way to accomplish this would be to reduce food intake while increasing exercise. You don't have to become a world-class athelete. But if you like swimming, or walking in the park, or Aikido, or any other form of physical activity, it can help you lose weight.

There will be other benefits, of course, including enjoying better health and feeling better about how you look. You'll have more resistance to winter colds and flu. You'll sleep more soundly and have better dreams. Besides all this, an exercise program will give you a magnificent opportunity to meet new people with similar interests. Perhaps the biggest plus will be your new-found control over your appetite. You'll be a lean jungle cat running on the edge of hungry.

Although the best plan to lose weight would include both diet and exercise, what if you truly want to lose weight but love to eat? The idea of actually cutting back on meals may be frightening. If so, there is still an answer in exercise.

Say you don't eat any more, or any less. But you do add an evening walk around the city park. Just by taking that little stroll daily, and doing nothing else, you could lose fourteen pounds in one year.

Now to the program. This diet helper asks how much you weigh, how much you want to weigh, and how many days you want to take to reach that target weight. After you fill in your vital information, the TV screen will display the total number of calories you'll need to burn off. Don't be dismayed. Even a couple of pounds seems to equal an astronomical number of calories.

The computer takes those calories, matches them up with the number of days until the target weight will be reached, and spells out how many calories per day you'll need to cut back to reach your goal.

Take those calories, mix diet and exercise to lose them, and soon you'll weigh in at fighting trim.

One word of caution. Don't be in a hurry to lose all the weight all at once. Two pounds or so a week is a sensible loss. Conduct a Spartan bootcamp for yourself, and discouragement is likely to set in. Take is easy on yourself and enjoy life—fit, trim, and healthy.

How the Program Works

Line 110 blanks the screen.

Line 120 positions the cursor in the top middle of the screen.

Line 130 prints the program title.

Line 160 inputs the variable CURRENT.

Line 190 inputs the variable DESIRED.

Line 210 inputs the variable DAYS.

Line 230 calculates how many pounds you want to lose.

Line 250 converts pounds to calories.

Lines 270 through 290 calculate how many calories need to be burned up a day in order to lose the weight during the specified period of time.

Lines 300 through 350 print the results.

String and Numeric Variables

CURRENT	Tells the computer how much you now weigh.
DESIRED	Tells the computer how much you ought to weigh.
DAYS	Tells how long you're allowing to remove those extra pounds.
LOSE	The result of subtracting how much you want to weigh by what you now weigh.
CALORIES	How many calories you need to burn up.

DAILYLOSS	How many calories you need to burn up each day.

The Program

```
100 REM WEIGHT LOSS/CALORIE TALLY
110 PRINT CHR$(125):REM CLEAR SCREEN
120 POSITION 10,2:REM LOCATE NEXT PRINT
130 PRINT "WEIGHT WATCHER"
140 PRINT :PRINT
150 REM GET WEIGHT
160 PRINT "CURRENT WEIGHT";:INPUT CURRENT
170 PRINT :PRINT
180 REM GET GOAL
190 PRINT "DESIRED WEIGHT";:DESIRED
200 PRINT :PRINT
210 PRINT "DAYS UNTIL TARGET WEIGHT";:INPUT DAYS
220 REM FIGURE POUNDS NEEDED TO LOSE
230 LOSE=CURRENT-DESIRED
240 REM CONVERT TO CALORIES
250 CALORIES=LOSE*3500
260 REM CONVERT TO CALORIES NEED TO LOSE DAILY
270 DAILYLOSS=CALORIES/DAYS
280 REM CONVERT TO INTEGER
290 DAILYLOSS = INT(DAILYLOSS +.5)
300 PRINT :PRINT
310 PRINT "CALORIES TO LOSE:"
320 POSITION 25,18
330 PRINT "TOTAL ";CALORIES
340 POSITION 25,20
350 PRINT "PER DAY ";DAILYLOSS
```

5 / Blood Alcohol Test

This program can give you an idea of where you stand, or wobble, legally. If you're planning an evening of social drinking, keep in mind that drunk driving laws are getting tougher. To find out if you will be bending the law, first plug in your vital statistics and see if you pass. If you go over the legal limit, the program will display a message telling you so.

It's true the capacity for alcohol varies from person to person. How well we maintain ourselves depends on our weight, body chemistry, and how fast and what we drink. Also important is whether our stomach is full or empty.

First, RUN the program, then enter your weight and how many ounces of alcohol you've imbibed. Next, enter the proof of the drink. This, by the way, is the only complicated part of the procedure. Straight whiskey, for example, might be labeled 86 or 100 proof. Proof is twice the alcoholic percentage. Therefore, 3.2 beer is 6.4 proof, 12 percent wine is 24 proof, and a highball, even though made up of a 100-proof vodka, won't be a 100-proof drink. If it's a nine-ounce drink, mixed with one ounce of vodka and eight ounces of orange juice, it's about 12 proof. That information should help in computing the actual proof of what you're drinking.

The result of all this computation will represent the approximate blood/alcohol level in your bloodstream. Each state has a specific blood/alcohol level that determines whether or not you are legally drunk. Line 230 in the program uses .01 (percent) as the legal limit. If you don't know your state's legal limit, a phone call to the state police will get you the answer.

How the Program Works

Line 110 blanks the screen.

Line 130 inputs your weight in pounds.

Line 150 inputs ounces of the drink.

Line 170 inputs the actual proof of the beverage.

Line 180 calculates the percentage of alcohol in your bloodstream.

Line 190 converts the percentage to a two-place decimal.

Line 210 prints the blood/alcohol level.

Line 230 checks to see whether or not you are legally drunk; if so, then prints LEGALLY DRUNK.

String and Numeric Variables

WEIGHT	Supplies the drunk/sober formula with your body weight in pounds.
OUNCES	Ounces imbibed.
PROOF	Actual proof of drink, mixed or straight.
CONCENTRATION	How much alcohol is in your bloodstream.
C	Converts CONCENTRATION to a two-place decimal.

The Program

```
100 REM BLOOD/ALCOHOL LEVEL MEASURE
110 PRINT CHR$(125):REM CLEAR SCREEN
120 PRINT
130 PRINT "HOW MUCH DO YOU WEIGH";:INPUT WEIGHT
140 PRINT
150 PRINT "HOW MANY OUNCES OF ALCOHOL";:INPUT OUNCES
160 PRINT
170 PRINT "PROOF OF ALCOHOL";:INPUT PROOF
180 LET CONCENTRATION=(OUNCES*PROOF*0.037)/WEIGHT
190 C=INT((CONCENTRATION * 100)+.5)/100
200 PRINT
210 PRINT "BLOOD LEVEL = ";C;PERCENT
220 PRINT
230 IF C>0.01 THEN PRINT "LEGALLY DRUNK"
```

6 / The ATARI Home Computer Crypto-System

Most crypto-systems create lines of text, broken down into five-character mystery words, that look exactly like what they are—garbled text. The scenario looks suspicious from first sight. There's little doubt that there's a secret lurking in those cipher text shadows, a dark secret the owner doesn't want you to know. For micro whiz-kids and business spooks it creates an instant challenge.

The beauty and truth of this crypto scheme is that it creates something quite unlike other crypto-systems. If someone stumbles across a sacred file, all that's seen is a computer program. Just program lines, GOSUBS, PRINTS, and DATA statements. But what happens if they LOAD the program and try to RUN? Simple. The program runs, giving them a very pleasing spectacle of what kinds of colors, bells, and whistles the ATARI Home Computer can conjure up.

Nothing mysterious here. And better yet, since it doesn't look like a secret code, no one tries to decipher the message.

Hiding secrets goes back to the old days, to ancient Greece, in fact, where slaves' heads were shaved, and secret messages tattoed on their scalps. Once the hair bristles were sufficient to hide the tattoed secrets, the slaves were dispatched to their destinies at the battle front. The technique of concealing messages is called *steganography*.

Cryptography is another science handed down from ancient Greece. *Crypto* means hidden and *graphy* means writing. Therefore, cryptography is the science of secret writing, the purpose of which is to prevent or delay an enemy or unauthorized person from obtaining intelligence by reading intercepted communications. Whew! In '80s lingo, that means enforcing your right to privacy.

To put both secret crafts into practice, first write the message in plain text, then read it back to yourself. Does it say exactly what you want it to say, in as few words as possible? Use abbreviations whenever possible, but only when the meaning is unmistakable. Use single words in place of whole sentences. Minimize the use of articles such as *a*, *an*, and *the*.

In secret message writing, do not use punctuation. Do not use the expression *stop*. Instead, use the abbreviations *ques* for question mark, *paren* for a parenthetical expression, *pd* for period, *cmm* for comma, and *quote/unquote* instead of quotation marks. Repetition leads to broken code. Do not repeat.

Further, numbers should be spelled out, i.e., *seven, nine, three.*

With the clear text message ready for encryption, RUN the program. It will ask for a code word. The standard admonitions apply in choosing a code word for encryption or time-sharing: Don't use your name, your lover's name, street address, or anything obvious. Don't make it easy for a spy to compromise your secrets.

Password entered, the computer will do a simple cipher substitution encryption, translating your clear text into cipher text and then translating the cipher text into data statements. Those data statements are stored in a cassette file from line 5000 on, until all the information you enter is encrypted. Each of the data numbers is actually an encrypted clear text character.

That cassette file can be read and the data statements decrypted, but only if you know the password. A stranger using a READ program will simply see data statements with groups of numbers. It will look like a computer program. For that reason, it helps to disguise the encrypted message to label the cassette tape as: DATA FOR LIGHT & SOUND SHOW.

The light and sound show program will read the DATA statements and use them to drive a flashy show. That way, if anyone loads the tape into memory and runs it, a light and sound show will provide delight for hours or until some other intrigue intervenes.

How the Program Works (Encrypt)

Line 120 blanks the screen.

Lines 130 and 140 dimension the string variables.

Line 150 assigns the value 5000 to the variable LINE.

Lines 160 and 170 tint the screen green.

Line 180 offers a choice between encryption and decryption.

Line 190 sends any answer but "1" to the decrypt routine.

Line 200 asks for a code word to be used in the encryption process.

Line 210 erases the screen, then positions the cursor nine spaces from the left.

Lines 220 through 280 print the encryption instructions.

Line 300 opens the cassette channel to receive encrypted data.

Line 310 sends the program off to the cassette file initialization routine. More on that later.

Line 320 blanks the screen.

Line 330 prompts the user to enter a clear text line.

Line 350 takes a single character, a word, or a sentence, up to a maximum of thirty-nine characters.

Line 370 looks for a carriage return. If the user enters a carriage return without any preceding text, it's the signal to quit encrypting.

Lines 390 through 520 do the encrypting. Line 390 starts the loop.

Line 440 will take each character in SENTENCE$ and convert it to its ATASCII number. The ATASCII number is simply a number the computer understands. For instance, it thinks of the letter "A" as the number "65".

Line 440, the ATASCII number is added to the code word's ATASCII number to disguise it.

Lines 450 through 510 take each encrypted number, and write them into the string variable C$. In essence, C$ is SENTENCE$ encrypted.

Line 540 sends C$ to the cassette file as a data statement. Remember the variable LINE from line 150. This data statement uses LINE to define its line number.

Line 560 increments LINE (originally 5000) by ten every time another data statement is written. What's really happening is we are writing a computer program made up of data statements and saving it to tape.

Line 580 branches back for the next SENTENCE$ to be encrypted.

Line 370 looks for a carriage return. When one is found, we go to line 600 where the cassette file is closed. Finished, line 620 runs the program to put us back at the menu.

Remember, in the beginning, at line 310, we gosub'd to line 630 for a cassette file initialization? That's because cassette files need a header, 128 bytes of dummy information written at the very beginning. Line 640 writes the dummy data. Line 650 returns us to the main program.

How the Program Works (Decrypt)

Line 660 blanks the screen.

Line 670 asks for the code word used to encrypt the message.

Line 680 blanks the screen.

Lines 690 through 750 print decryption instructions.

Line 760 is an error trap; It looks for errors. In this case, it watches for the end of the cassette file, once we start loading it.

Line 770 opens the cassette channel for a read.

Line 780 blanks the screen.

Line 800 reads each BYTE from the cassette file, one by one.

Line 830 decrypts the BYTE.

Line 840 converts the decrypted character to its string variable equivalent. Remember, we said the computer thought of the letter "A" as the number 65. Well, if the decrypted BYTE was 65, line 840 would convert BYTE to BYTE$, and BYTE$ would equal "A".

Line 860 sends us back for the next BYTE.

Line 880 halts the computer, once the error trap finds the last BYTE in the cassette file. Press any key, and we go on.

String and Numeric Variables

CODE$	The code word used in the encryption formula.
SENTENCE$	The clear text character, word, or sentence to be encrypted.
BYTE$	The individual string variable from SENTENCE$.
BYTE	Holds the ATASCII number for BYTE$.
A$	A buffer used to concatenate encrypted BYTE$s.
B$	Consists of a comma and a space. Separates the numbers in each DATA statement.
C$	A buffer used to concatenate encrypted BYTE$s.
WHICH	Used with menu to choose ENCRYPT or DECRYPT.

LENGTH	Length of SENTENCE$. Loop back until each character is encrypted.
COUNT	Bookkeeper; tells which character in string is being encrypted.
LINE	Assigns line number of new DATA statement.

The Program

```
100 REM CRYPTO-SYSTEM
110 REM USE WITH LIGHT SHOW PROGRAM
120 PRINT CHR$(125):REM CLEAR SCREEN
130 DIM CODE$(25):DIM SENTENCE$(39)
140 DIM BYTE$(3),A$(40),B$(40),C$(255)
150 LINE=5000:REM BEGIN LINE # DATA STATEMENTS
160 SETCOLOR 1,14,4:REM GREEN SCREEN FOR EYES
170 SETCOLOR 2,14,0:REM MORE GREEN
180 PRINT :PRINT "ENCRYPT, OR DECRYPT  (1 OR 2)";:INPUT WHICH
190 IF WHICH<>1 THEN GOTO 660
200 PRINT :PRINT "CODE WORD =  ";:INPUT CODE$
210 PRINT CHR$(125):POKE 85,9
220 PRINT "ABOUT TO ENCRYPT"
230 PRINT :PRINT "1.  INSERT A BLANK TAPE"
240 PRINT :PRINT "2.  SET RECORDER COUNTER TO 0"
250 PRINT :PRINT "3.  PRESS   RECORD/PLAY"
260 PRINT :PRINT "4.  HIT RETURN"
270 PRINT :PRINT "5.  THEN WAIT FOR PROMPT . . ."
280 POKE 755,0:REM TO HIDE CURSOR
290 REM OPEN CASSETTE FOR WRITE
300 OPEN #1,8,0,"C:"
310 GOSUB 630:REM GO INITIALIZE
320 PRINT CHR$(125):REM CLEAR SCREEN
330 PRINT :PRINT "WORD(S) TO ENCRYPT  (39 CHARS. LINE)"
340 PRINT :PRINT
350 INPUT SENTENCE$:REM GET LINE CLEAR TEXT
360 REM ON CARRIAGE RETURN GOTO 590
370 IF SENTENCE$="" THEN GOTO 590
380 REM GET LENGTH OF LINE
390 FOR LENGTH=1 TO LEN(SENTENCE$)
400 COUNT=COUNT+1
410 REM TAKE APART LETTER BY LETTER
420 LET BYTE$=SENTENCE$(COUNT,LENGTH)
430 REM ENCRYPT LETTER BY LETTER
440 LET BYTE=ASC(BYTE$)+ASC(CODE$)
450 B$=", ":REM B$ = COMMA THEN SPACE
460 REM ON FIRST DATA NUMBER FORGET COMMA
470 IF COUNT=1 THEN B$=" "
480 A$=STR$(BYTE)
490 REM ADD COMMA TO ENCRYPTED LETTER
500 B$(LEN(B$)+1)=A$
510 C$(LEN(C$)+1)=B$
520 NEXT LENGTH
530 REM WRITE LINE TO CASSETTE AS DATA STATEMENT
540 PRINT #1;LINE;" DATA ";C$;:PRINT #1
550 C$="":REM BLANK OUT STRING
560 LINE=LINE+10:REM INCREMENT LINE #
570 COUNT=0:LENGTH=0:I=0
```

```
580 GOTO 350:REM GET NEXT CLEAR TEXT LINE
590 REM IF DONE ENCRYPTING CLOSE FILE
600 CLOSE #1
610 REM NOW RETURN TO MENU
620 RUN
630 REM INITIALIZE CASSETE FILE FOR WRITE
640 FOR I=1 TO 128:PRINT #1; " ";:NEXT I:PRINT #1
650 RETURN
660 PRINT CHR$(125):REM CLEAR SCREEN
670 PRINT :PRINT "CODE WORD =  ";:INPUT CODE$
680 PRINT CHR$(125):REM CLEAR SCREEN
690 POSITION 8,2:REM LOCATE CURSOR
700 PRINT "ABOUT TO DECRYPT"
710 PRINT :PRINT "1.  INSERT REWOUND ENCRYPTED TAPE"
720 PRINT :PRINT "2.  SET RECORDER COUNT TO ZERO"
730 PRINT :PRINT "3.  PRESS PLAY BUTTON"
740 PRINT :PRINT "4.  HIT RETURN"
750 PRINT :PRINT "5.  WHEN TAPE STOPS, TYPE: 'GOTO 780'"
760 TRAP 870
770 ENTER "C:"
780 PRINT CHR$(125):REM CLEAR SCREEN
790 REM READ DATA STATEMENTS
800 READ BYTE
810 TRAP 870
820 REM DECRYPT CHARACTER BY CHARACTER
830 LET BYTE=BYTE-ASC(CODE$)
840 LET BYTE$=CHR$(BYTE)
850 PRINT BYTE$
860 GOTO 800
870 POSITION 0,21
880 PRINT "HIT RETURN FOR NEXT TASK";:INPUT A$
890 RUN
```

Light and Sound Show

The light show draws a big glob of color on the screen, thanks to Player/Missile graphics, and moves it around the screen, accompanied by its very own ballad. While it's meant to be driven by data generated by the crypto program, it's perfectly feasible to write your own set of values. However, only programmers with graphics experience will do this easily. Draw a shape, if you thoroughly understand the intricacies of Player/Missile graphics, and store it in a DATA line. Choose the musical notes you wish played, and intersperse them with the move numbers. By the way, 120 represents the middle of the TV screen. Try a few different dazzling variations in this Personal Computer art form.

How the Program Works

Line 110 blanks the screen.

Lines 140 through 220 print the instructions.

Line 240 turns on the cassette recorder and loads the DATA statements. These are really program lines, starting at line 5000, and continuing for as many lines as there were clear text lines. These lines will load below the Light Show program and become a part of it. If you save the program after loading the DATA statements, they become part of the program.

Line 630 watches for the end of DATA. When encountered, program control branches to line 630.

Line 270 switches on Graphics Mode 3, with a full screen, no text window.

Lines 290 and 300 calculate where to draw "Player/Missile" graphics.

Line 290 gets the top of available RAM and stores that location where the program can find it.

Line 300 takes that stored number and figures out where to draw the Player/Missile graphics. That location is stored in the variable, PMBASE.

Line 310 turns P/M graphics on.

Line 320 lends us a variable, DRAW, to actually begin drawing our player. Lines 350 through 430 draw that "player" forty lines tall. Each line is eight boxes wide, and made up of empty or filled boxes.

Line 370 reads the DATA, a 255 would print a solid line, 0 would be a blank line.

Line 390 increments the variable DRAW each time through the loop. DRAW, as you recall, is the spot in memory where we began to draw. Incrementing DRAW increments the draw location.

Line 410 pokes the draw informtion at memory location DRAW.

Line 430 loops back forty times, until the player is drawn.

Line 450 reads the DATA number.

Line 470 loads that number in the player color register, painting it that color.

Line 490 gets a random number between 0 and 20. Line 520 multiplies it by 10. That gives us a range of 10 to 200. We need that range to be able to move the player whose horizontal position needs to be between about 20 and 250.

Line 540 assigns that number to the player's horizontal position register.

Line 560 reads the next DATA number. Line 580 plays that number as a musical note.

Line 600 plays (delays) that note for the count, 1 to 100.

Line 620 sends us back for more DATA reads.

Line 650 resets the TRAP. In this case we use the TRAP in line 250 to watch for the end of DATA, in other words, once we've read all the DATA numbers once. When that happens, the program jumps down to line 630. The RESTORE command lets us read the DATA again, to keep the program running. The TRAP 630 command in line 650 resets the trap so it knows to RESTORE again, every time we have read all the DATA.

String and Numeric Variables

TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player-Missile memory.
DRAW	Where in memory to draw player number 1.
LOOP	Loop counter.
COLOR	Read DATA, specify player color.
RANDUM	Random Integer Number.
TONE	Read DATA, specify musical note.
DELAY	Time TONE is played.

The Program

```
100 REM DUMMY LIGHT SHOW
110 PRINT CHR$(125):REM CLEAR SCREEN
120 REM LOCATE CURSOR
130 POSITION 7,3
140 PRINT "LIGHT AND SOUND SHOW!"
150 POKE 84,5
160 PRINT "1.  PLACE TAPE WITH DATA IN RECORDER"
170 POKE 84,7:REM SET MARGIN
180 PRINT "2.  PUSH PLAY.  HIT RETURN"
190 POKE 84,9:REM SET MARGIN
200 PRINT "3.  WHEN TAPE STOPS LOADING DATA,"
```

```
210 POKE 84,11:REM SET MARGIN
220 PRINT "4.  TYPE: 'GOTO 250'  "
230 REM LOAD DATA FROM 410 RECORDER
240 ENTER "C"
250 TRAP 630
260 REM CHOOSE GRAPHICS
270 GRAPHICS 3+16
280 REM INITIALIZE GRAPHICS
290 TOP=PEEK(106)-8:POKE 54279,TOP
300 PMBASE=256*TOP:POKE 559,46
310 POKE 53277,3
320 DRAW=PMBASE+540
330 POKE 53256,3
340 REM DRAW PLAYER 40 LINES TALL
350 FOR LOOP=1 TO 40
360 REM TAKE DATA
370 READ BYTE
380 REM KEEP TRACK HOW TALL
390 DRAW=DRAW+1
400 REM DRAW ONE LINE AT A TIME
410 POKE DRAW,BYTE
420 REM NEXT LINE TO DRAW
430 NEXT LOOP
440 REM NEXT DATA FOR COLOUR
450 READ COLOR
460 REM COLOR THE PLAYER
470 POKE 704,COLOR
480 REM CHOOSE RANDOM NUMBER
490 RANDUM=INT(20*RND(1))
500 REM MULTIPLY BY TEN
510 REM PUT THAT # IN HORIZ. POS
520 RANDUM=RANDUM*10
530 REM MOVE PLAYER AROUND SCREEN
540 POKE 53248,RANDUM
550 REM GET DATA
560 READ TONE
570 REM PLAY MUSICAL NOTES WITH DATA
580 SOUND 0,TONE,10,15
590 REM DELAY LOOP
600 FOR DELAY=1 TO 100:NEXT DELAY
610 REM GET MORE DATA
620 GOTO 440
630 REM WHEN DATA GONE RESTORE
640 REM THEN BACK TO PROGRAM
650 RESTORE :TRAP 630:GOTO 440
660 REM ENCRYPTED TEXT IS STORED
670 REM AS DATA STATEMENTS
680 REM STARTING AT LINE NUMBER 5000
```

7 / Medical History

Accurate medical records can be important for any family. They may come in very handy in case of sudden illness. If a family member suffers a severe allergic reaction, the emergency room staff may need to know the current medications the injured person has been taking. Will you be able to provide this information in the high stress environment of a trauma center? In another instance, you may need only to provide a printout of Grandma Shelly's file to make the doctor's job easier.

The personal computer will make the collecting and sorting of such medical information easy, through writing to cassette tape or disk. The following Medical Record program uses some of the very special characteristics of the ATARI 400/800 and 1200XL Home Computers. Namely, the video screen is live. That means anything displayed on the screen can be read by the ATARI Home Computer and sent wherever you want it to go. You'll soon understand more of what I mean.

At the start, you have a menu giving two choices.

1. Read an existing record.
2. Create a record.

Since we don't have any medical records yet, we'll ENTER a "2". That done, the screen clears and prints the Medical Form. At each question mark prompt, enter the appropriate information: Name, Date of Birth, Known Allergies, and so on. After the last bit of information is entered, the ATARI Home Computer will "beep-beep." That's the cue to:

1. Load a blank, rewound cassette.

2. Press the PLAY and RECORD buttons on the 410 Recorder.
3. Hit RETURN.
4. Wait, patiently, while the program saves the data to tape.

Once that's accomplished, the screen will clear and print the menu.

Do you want to check the file for accuracy, or update current medications? If so:

1. Rewind the cassette tape.
2. Press the PLAY button.
3. Enter menu selection "1".

The screen will clear and print the medical record form. When the computer beeps, hit RETURN. The recorder then turns on and loads the file. In a matter of moments, the medical history will fill in all the blanks. But things are much different at this point.

Remember we said the screen was live? You'll see a question mark prompt in front of the first character of the person's name. At this point, if you hit RETURN, that name will be saved exactly as is. If the name is misspelled, use the editing keys to correct it. Once you hit RETURN, whatever was printed on that line will be saved exactly as it was displayed on the screen.

When you hit RETURN, the question mark prompt jumps down to the Date of Birth line. Again, if you hit RETURN, the line will be saved exactly as it appeared on the screen. Add, delete, or leave as is each of the lines all the way down to Current Medications. Once you hit RETURN after Current Medications, you will hear the familiar beep-beep. If you want to save the updated file:

1. Rewind the tape to overwrite the old information.
2. Press the RECORD and PLAY buttons.
3. Hit RETURN.

How the Program Works

Lines 110 through 180 dimension the string variables.

Line 190 blanks the screen.

Lines 200 through 280 format and print the menu selection.

Line 300 opens the keyboard for a read. Just press any key; it's not necessary to hit RETURN.

Line 320 retrieves each key's ATASCII number, for that single key response, and stores it in the variable RESPONSE.

Line 340 switches the keyboard back to its normal mode.

Line 350 routes the program to the subroutine starting at line 950.

Lines 950 through 1140 format and print the medical form. This same form is used for both read and write.

Line 370 compares the job selection stored in RESPONSE. The number 1's ATASCII code is "49". If the USER presses "1" the program goes to line 680 and reads back a stored medical file. Any response other than a "1" will cause the program to continue on to line 380.

Lines 380 through 530 format the prompts which ask for Name, Birthdate, and so on. Each bit of information is sorted in its own string variable, e.g., Name in NAME$.

Line 550 opens the cassette channel for a write. One by one, each string variable is written to the tape. When done writing, line 650 closes the channel.

Line 670 sends us back to the menu.

Reading a Medical Record

Line 700 opens the cassette channel for a read.

Lines 710 through 910 read the information stored on the tape. The INPUT #1, WHATEVER$, reads each string variable, the POSITION commands locate the prompts.

Line 930 turns the cassette channel off. Line 940 sends us to line 400.

It's at this point the information can be changed. Those POSITION commands we mentioned in lines 710 through 910 place the old information on the right of the cursor.

String and Numeric Variables

NAME$	Retrieve or write person's name.
BIRTH$	Retrieve or write person's birthdate.
ALLERGY$	Retrieve or write known allergies.
VACCINATION$	Retrieve or write vaccinations, innoculations.
AILMENT$	Retrieve or write illnesses.
BONE$	Retrieve or write bone fractures.
MED$	Retrieve or write current medications and dosage.

TITLE$	MEDICAL RECORD.
RESPONSE	Tells menu choice; read or write record.

The Program

```
100 REM MEDICAL HISTORY
110 DIM NAME$(36)
120 DIM BIRTH$(28)
130 DIM ALLERGY$(36)
140 DIM VACCINATION$(36)
150 DIM AILMENT$(36)
160 DIM BONE$(36)
170 DIM MED$(36)
180 DIM TITLE$(14)
190 PRINT CHR$(125):REM CLEAR SCREEN
200 POKE 85,12:REM INDENT NEXT PRINT
210 TITLE$="MEDICAL RECORD"
220 PRINT TITLE$
230 PRINT :PRINT
240 PRINT "1.  READ RECORD"
250 PRINT :PRINT
260 PRINT "2.  WRITE RECORD"
270 PRINT :PRINT
280 PRINT "PLEASE CHOOSE  (1 OR 2)"
290 REM OPEN KEYBOARD FOR READ
300 OPEN #1,4,0,"K"
310 REM READ KEYBOARD
320 GET #1,RESPONSE
330 REM CLOSE KEYBOARD
340 CLOSE #1
350 GOSUB 950
360 REM GO DO JOB CHOSEN
370 IF RESPONSE=49 THEN GOTO 680
380 REM ANY RESPONSE OTHER THAN #1 AND PROGRAM CONTINUES
390 REM NOW WRITE MEDICAL HISTORY
400 POKE 84,3
410 INPUT NAME$
420 POSITION 16,5
430 INPUT BIRTH$
440 POKE 84,8
450 INPUT ALLERGY$
460 POKE 84,11
470 INPUT VACCINATION$
480 POKE 84,14
490 INPUT AILMENT$
500 POKE 84,17
510 INPUT BONE$
520 POKE 84,20
530 INPUT MED$
540 REM OPEN CHANNEL TO CASSETTE PLAYER
550 OPEN #1,8,0,"C:"
560 REM WRITE TO CASSETTE FILE
570 PRINT #1;NAME$
580 PRINT #1;BIRTH$
590 PRINT #1;ALLERGY$
600 PRINT #1;VACCINATION$
610 PRINT #1;AILMENT$
620 PRINT #1;BONE$
630 PRINT #1;MED$
```

```
640 REM CLOSE CHANNEL TO CASSETTE
650 CLOSE #1
660 REM RETURN TO MENU FOR NEXT JOB
670 GOTO 190
680 REM READ MEDICAL RECORDS
690 REM OPEN CASSETTE FOR READ
700 OPEN #1,4,0,"C:"
710 INPUT #1,NAME$
720 POSITION 3,3
730 PRINT NAME$
740 INPUT #1,BIRTH$
750 POSITION 17,5
760 PRINT BIRTH$
770 INPUT #1,ALLERGY$
780 POSITION 3,8
790 PRINT ALLERGY$
800 INPUT #1,VACCINATION$
810 POSITION 3,11
820 PRINT VACCINATION$
830 INPUT #1,AILMENT$
840 POSITION 3,14
850 PRINT AILMENT$
860 INPUT #1,BONE$
870 POSITION 3,17
880 PRINT BONE$
890 INPUT #1,MED$
900 POSITION 3,20
910 PRINT MED$
920 REM CLOSE CHANNEL TO RECORDER
930 CLOSE #1
940 GOTO 400
950 REM PRINT MEDICAL FORM
960 PRINT CHR$(125):REM CLEAR SCREEN
970 POSITION 12,0:REM WRITE AT TOP OF SCREEN
980 PRINT TITLE$
990 REM USE INVERSE VIDEO
1000 POKE 84,2
1010 PRINT "NAME: FIRST, MIDDLE, LAST"
1020 POKE 84,5
1030 PRINT "DATE OF BIRTH"
1040 POKE 84,7
1050 PRINT "ALLERGIES"
1060 POKE 84,10
1070 PRINT "VACCINATIONS"
1080 POKE 84,13
1090 PRINT "SERIOUS AILMENTS"
1100 POKE 84,16
1110 PRINT "BROKEN BONES"
1120 POKE 84,19
1130 PRINT "CURRENT MEDICATIONS"
1140 RETURN
```

8 / Ghost Town Vampire Girls

Diana and her vampire sister, Tiger-breath, have invited you to visit their ghostly retreat. Once at their castle, you will find a succession of double doors. Behind one door is the very charming Lady Diana; behind the other, her demented half-sister, the vampire. On your adventure through the passageways of their ghostly retreat, you must choose from a succession of doors. Follow Diana as she lures you toward her boudoir. But be careful. Her blood-thirsty sister is stalking you. So choose a door—and take a chance—on the lady, or the tiger.

To play, use the joystick to move the red square, representing you, to the door of your choice. Once you've moved close enough to one of the doors to count as a decision, the game's theme song will play. At the end of the tune, you'll know whether you've met your fate or are getting closer to your rendezvous with Lady Di.

As you pick your way from room to room, you'll find clues as to Diana's presence. Don't pick the wrong room, or Tiger-breath will get you. If she does, and you want to try another round of castle-touring, hit any key.

How the Program Works

Line 110 blanks the screen.

Line 120 assigns the line number of the subroutine MUSIC, where the theme song plays.

Line 130 defines the subroutine where the computer waits for you to press any key.

Line 140 selects graphics mode 2, full screen, no text window.

Line 150 paints the screen.

Lines 160 through 190 format and print the program title, while line 210 plays the theme song.

Line 220 chooses graphics mode 5, with a text window.

Line 230 gosubs to draw the big red box.

Lines 240 through 260 turn on the graphics and place the joystick-controlled player on the screen.

Lines 280 through 850 print the clues, room by room.

Lines 870 through 1080, once face to face with Diana, the player is moved off-screen to allow full screen text.

Line 1110 flips a coin to see whether you win or lose. Any number other than a "1" and you lose.

If You Lose . . .

Lines 1130 through 1250 print the lose message.

Line 1260 gosubs to the "Press Any Key" WAIT subroutine.

Line 1270 starts a new game.

If You've Won!

Line 1290 blanks the screen.

Lines 1300 through 1360 print the victory message. Line 1370 plays the theme song.

Drawing the Double Doors and Player

Line 1570 changes the color of the doors each turn; line 1580 does the actual painting.

Lines 1590 through 1660 draw and fill both doors.

Lines 1680 through 1770 move the player (the big red box).

Setting the Odds

Line 1790 sets the win/lose odds. A random number between 0 and 12 is chosen. Your chances are 1 out of 13 to lose. Change the magic number to 24, and the odds of success turn in your favor, to 25 to 1. But drop the number to 3 and you have only a 1-in-4 chance of findinng the lucky lady.

Player Missile Graphics

Lines 1850 and 1860 allocate the Player/Missile memory.

Lines 1880 and 1890 poke zeros into the memory locations where the box will be drawn. That dumps any unwanted in-

formation left from any other program you may have run since firing up the computer.

Lines 1910 through 1930 put a solid line in six of the memory locations we just filled with zeros. This draws a little box six lines tall, in the center of the screen.

The WAIT subroutine

Lines 1950 through 1980 halt the program while it waits for a key to be pressed—any key, other than BREAK or the function keys. Until one of the keys is pressed, the program loops at line 1970, patiently waiting for you to do something.

MUSIC Subroutine

Line 2010 is a loop, it will read a data number and place it in the variable TONE. The SOUND command will play TONE as a musical note.

Line 2020 will delay while the variable DELAY counts from 1 to 40. It's during this delay that the musical note TONE plays. Increase the delay from 40 to 100 and the note plays for a longer duration. Decrease the delay, say from 40 to 10, and the note plays for a shorter period of time, almost a pip. Following the DELAY loop, the NEXT LOOP sends us back for the next TONE.

String and Numeric Variables

MUSIC	Line number where music subroutine is located.
WAIT	Line number where "Press Any Key To Go On" subroutine is located.
RED	Horizontal position of red box.
A	Joystick reading.
PAINT	Which color to paint double door.
DELAY	Length of delay in music routine.
TONE	Pitch of sound in falling subroutine (lines 1480 & 1490).
RANDUM	Random number to select win/lose odds.
TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.

MEM	Temporary location in PM memory where zeros are drawn to blank old players.

The Program

```
100 REM GHOST TOWN VAMPIRE GIRLS GAME
110 PRINT CHR$(125):REM CLEAR SCREEN
120 MUSIC=2010
130 WAIT=1950
140 GRAPHICS 2+16:REM CHOOSE GRAPHICS
150 SETCOLOR 4,8,2:REM BACKGROUND COLOR
160 POKE 85,4:REM SET HORIZ. MARGIN
170 PRINT #6;"GHOST TOWN"
180 POSITION 4,8
190 PRINT #6;"VAMPIRE GIRLS"
200 REM PLAY MUSIC
210 GOSUB MUSIC
220 GRAPHICS 5
230 GOSUB 1840:REM GO DRAW PLAYER
240 POKE 53277,3:REM TURN ON GRAPHICS
250 POKE 704,48:REM COLOR PLAYER RED
260 POKE 53248,120:REM PUT PLAYER IN CENTER OF SCREEN
270 REM BEGIN THE GAME
280 REM PRINT THE FIRST MESSAGE
290 PRINT "ENTER THE CASTLE KEEP. THE DOOR'S NOT"
300 PRINT "LOCKED. YOU CAN FAINTLY HEAR HER SING"
310 PRINT "A BAWDY LOVE BALLAD.  ENTER."
320 REM GO DRAW DOORS
330 GOSUB 1560
340 REM SECOND MESSAGE
350 PRINT "SAFE INSIDE, ANOTHER SET OF DOORS. "
360 PRINT "HER CLOAK LIES ON THE STONE FLOOR"
370 PRINT "STILL ALIVE WITH THE SCENT OF HER"
380 REM GO DRAW DOORS
390 GOSUB 1560
400 PRINT "THE TROPHY ROOM. STAG AND DEER HEADS "
410 PRINT "MOUNTED OVER THE CRACKLING FIREPLACE."
420 PRINT "A SILK BLOUSE RESTS ON A BEARSKIN RUG"
430 GOSUB 1560
440 PRINT "THE DEN, WHERE SHE COUNTS GOLD COINS,"
450 PRINT "TOYS WITH HANDFULS OF RUBIES AND "
460 PRINT "SAPPHIRES.  HER VOICE SOUNDS CLOSER."
470 GOSUB 1560
480 PRINT "A ROSE GARDEN IN BLOOM. THE SKULL AND"
490 PRINT "BLEACHED BONES OF AN UNLUCKY SUITOR."
500 PRINT "WILL YOU BE THE NEXT TO DIE?"
510 GOSUB 1560
520 PRINT "FEASTING HALL LINED WITH THE SHIELDS "
530 PRINT "OF BRAVE KNIGHTS WHO DIED TRYING TO"
540 PRINT "SATISFY HER WICKED THIRST. . ."
550 GOSUB 1560
560 PRINT "'JUST BEYOND THE DOOR,' SHE PURRS..."
570 PRINT "'COME PLEASE ME. JUST ONE MORE DOOR."
580 PRINT "'DON'T BE AFRAID OF THAT NASTY TIGER'"
590 GOSUB 1560
600 PRINT "THE BEDROOM. EMPTY. SHE LIED."
610 PRINT "WOMEN ARE PRONE TO THAT YOU KNOW. BUT"
620 PRINT "THERE IS ANOTHER SET OF DOORS,THOUGH."
630 GOSUB 1560
```

```
640 PRINT "A SECRET PASSAGE TO THE WINE CELLAR."
650 PRINT "CASKS OF HEARTY RED WINE. RATS SCURRY"
660 PRINT "AT YOUR FEET, MOCKING YOU."
670 GOSUB 1560
680 PRINT "THE DUNGEON. AN OLD BEGGAR IN RAGS"
690 PRINT "CACKLES. POINTS A GRIMY FINGER AT YOU"
700 PRINT "'ANOTHER OF HER LOVER-BOYS, EH?"
710 GOSUB 1560
720 PRINT "SHARPEN YOUR SWORD?  HOW DOES ONE"
730 PRINT "COOK TIGER MEAT?"
740 PRINT "SEARCH THE TOWER. LOOK BY THE WINDOW."
750 PRINT "HER LONG WHITE DRESS LEFT AS A CLUE."
760 PRINT "MADAME PLAYS HARD TO GET."
770 GOSUB 1560
780 PRINT "A COOK FIRE ROARS IN THE KITCHEN, A"
790 PRINT "ROAST SPUTTERS ON THE SPIT. DINNER"
800 PRINT "FOR TWO?"
810 GOSUB 1560
820 PRINT "STEP OVER THE DEAD WARRIOR. DIANA HAS"
830 PRINT "KILLED MORE MEN THAN THE BLACK KNIGHT"
840 PRINT "OF WODIN-BAGEN.  PRESS ON."
850 GOSUB 1560
860 REM MOVE PLAYER OFF-SCREEN
870 POKE 53248,1
880 GRAPHICS 5+16
890 REM PLAYER FINDS DIANA
900 PRINT "CONGRATULATIONS . . ."
910 PRINT
920 PRINT "YOU ARE FACE TO FACE WITH DIANA"
930 PRINT "THE SEDUCTRESS, THE MURDERESS OF "
940 PRINT "SCORES OF BRAVE MEN."
950 PRINT
960 PRINT "LOOK AT HER SMILE.  OR IS IT AN EVIL"
970 PRINT "GRIN?"
980 PRINT
990 PRINT "SLOWLY, SEDUCTIVELY, SHE LETS THE "
1000 PRINT "REMNANTS OF HER GARMENTS FALL TO THE"
1010 PRINT "FLOOR."
1020 PRINT
1030 PRINT "WHAT DOES HER HALF-GRIN, HALF-SMILE"
1040 PRINT "MEAN?"
1050 PRINT
1060 PRINT "HER LIPS PART AS YOU DRAW NEAR. . ."
1070 PRINT
1080 PRINT "PRESS ANY KEY TO FIND OUT"
1090 GOSUB WAIT
1100 REM TAKE CHANCE ON GOOD LUCK
1110 RANDUM=INT(2*RND(1)):IF RANDUM=1 THEN 1290
1120 REM YOU LOSE
1130 PRINT CHR$(125):REM CLEAR SCREEN
1140 PRINT :PRINT
1150 PRINT "WHAT'S THIS!"
1160 PRINT
1170 PRINT "A RAZOR SHARP BLADE LOCKED BETWEEN"
1180 PRINT "HER CLENCHED TEETH!"
1190 PRINT
1200 PRINT "YOU FEEL THE SAVAGE BLADE RIPPING"
1210 PRINT "ACROSS YOUR THROAT. JUGULAR VEIN "
1220 PRINT "PUMPING HOT BLOOD, AS YOU REALIZE YOU"
1230 PRINT "ARE ABOUT TO DIE . . ."
1240 PRINT
1250 PRINT "IT'S OVER.  SORRY OLD FELLOW."
1260 GOSUB WAIT
1270 GOTO 220
```

```
1280 REM YOU WIN
1290 PRINT CHR$(125):REM CLEAR SCREEN
1300 PRINT
1310 PRINT "QUIVERING IN ANTICIPATION . . ."
1320 PRINT
1330 PRINT "SHE TREMBLES, AS YOU BEGIN TO EXPLORE"
1340 PRINT "HER TENDER SECRETS."
1350 PRINT
1360 PRINT "SUCCESS!"
1370 GOSUB MUSIC
1380 POKE 84,20
1390 PRINT "IF YOU WANT TO TRY AGAIN, HIT ANY KEY"
1400 GOSUB WAIT
1410 GOTO 220
1420 POKE 53248,1:REM MOVE PLAYER OFF-SCREEN
1430 REM YOU MEET UP WITH VAMPIRE GIRL
1440 GRAPHICS 2:SETCOLOR 4,8,2:SETCOLOR 2,3,2
1450 REM PAINT SCREEN BORDER RED
1460 SETCOLOR 0,4,14:SETCOLOR 3,3,1
1470 REM CREATE FALLING SOUND
1480 FOR TONE=45 TO 75:SOUND 0,TONE,10,10
1490 FOR DELAY=1 TO 10:NEXT DELAY:NEXT TONE:SOUND 0,0,0,0
1500 PRINT #6;"    GROWL"
1510 PRINT #6;" SNARF - CHOMP"
1520 PRINT "YOU'VE BEEN GOBBLED BY A VAMPIRE GIRL"
1530 PRINT " (HIT ANY KEY TO PLAY AGAIN)"
1540 GOSUB WAIT
1550 GOTO 220
1560 REM DRAW AND PAINT BOTH DOORS
1570 PAINT=PAINT+1:IF PAINT>3 THEN PAINT=1:IF PAINT=0 THEN PAINT=PAINT+
1
1580 COLOR PAINT
1590 PLOT 10,5:DRAWTO 10,30:DRAWTO 25,30
1600 DRAWTO 25,5:DRAWTO 10,5:COLOR PAINT +1:IF PAINT =0 THEN PAINT = 1
1610 PLOT 45,5:DRAWTO 45,30:DRAWTO 60,30
1620 DRAWTO 60,5:DRAWTO 45,5
1630 COLOR 3:PLOT 24,29:DRAWTO 24,6:DRAWTO 11,6:POSITION 11,29
1640 POKE 765,3:XIO 18,#6,0,0,"S"
1650 COLOR 3:PLOT 59,29:DRAWTO 59,6:DRAWTO 46,6:POSITION 46,29
1660 POKE 765,3:XIO 18,#6,0,0,"S"
1670 POKE 755,0:REM HIDE CURSOR
1680 RED=120
1690 POKE 53248,RED:REM LOCATE PLAYER
1700 LET A=STICK(0)
1710 REM MOVE PLAYER
1720 IF A=11 THEN RED=RED-5:POKE 53248,RED
1730 IF A=7 THEN RED=RED+5:POKE 53248,RED
1740 REM ONCE DOOR CHOOSEN MOVE
1750 IF RED<81 THEN GOTO 1790
1760 IF RED>149 THEN GOTO 1790
1770 GOTO 1700
1780 REM CHOOSE ODDS FOR GAME
1790 LET RANDUM=INT(12*RND(1))
1800 REM IF RANDUM = 5 YOU LOSE
1810 IF RANDUM=5 THEN GOSUB 1420
1820 GOSUB MUSIC
1830 RETURN
1840 REM GET TOP OF RAM ADDRESS
1850 TOP=PEEK(106)-8
1860 POKE 54279,TOP:PMBASE=256*TOP:POKE 559,46
1870 REM CLEAR MEMORY WITH ZEROS
1880 FOR MEM=PMBASE+512 TO PMBASE+640
1890 POKE MEM,0:NEXT MEM
1900 REM DRAW PLAYER
```

```
1910 POKE PMBASE+560,255:POKE PMBASE+561,255
1920 POKE PMBASE+562,255:POKE PMBASE+563,255
1930 POKE PMBASE+564,255:POKE PMBASE+565,255
1940 RETURN
1950 POKE 764,255
1960 POKE 755,0:REM HIDE CURSOR
1970 IF PEEK(764)=255 THEN 1970
1980 POKE 764,255
1990 RETURN
2000 REM PLAY MUSIC ROUTINE
2010 FOR LOOP=1 TO 16:READ TONE:SOUND 0,TONE,10,8
2020 FOR DELAY=1 TO 40:NEXT DELAY:SOUND 0,0,10,8:NEXT LOOP
2030 RESTORE :REM RESTORE TO PLAY NOTES AGAIN
2040 RETURN :REM MUSIC STORED AS DATA
2050 DATA 27,32,55,82,36,27,54,41,36
2060 DATA 54,36,54,32,41,29,61
```

9 / Beowulf versus Grendel

Beowulf the warrior-man meets Grendel the monster on the field of battle. You are heroic Beowulf. Hear the trumpets blare! Grendel is the hideously ugly monster. He has been terrorizing the village, killing cattle, torching homes and barns, stealing beautiful maidens, and murdering good men in cold blood.

You, brave fellow that you are, have volunteered to slay this evil monster Grendel.

Grendel is the red player; you, the yellow. Use the joystick to maneuver across the field. Push the trigger button to slash out with your light saber. The first warrior to make ten hits on the other wins.

To play another game, press any key.

How the Program Works

Line 110 blanks the screen.

Lines 120 and 130 tell where the subroutines begin.

Line 140 chooses Graphics Mode 2, so we can print in tall letters. SETCOLOR paints the screen.

Lines 160 through 180 format and print the game title.

Line 190 branches to play the theme song.

Lines 200 to 240 prompt the player for whether or not he or she needs instructions.

Line 270 chooses a random number between 0 and 9. Lines 290

through 380 take the random number and give it a plus or minus value, then assign that value to the variable POS.

Line 400 takes the value stored in POS and adds it to Grendel's current position (RED). If POS is a negative number, Grendel moves to the left; if positive, to the right.

Lines 410 through 450 compare Beowulf's horizontal position (YELLOW), to Grendel's. If the difference is more than 15, move Grendel closer to Beowulf.

Line 470 locates a "missile" (Player/Missile) at Grendel's horizontal position.

Lines 490 and 5500 draw Grendel's light saber and create its sound.

Line 510 holds that sound for thirty microcomputer beats.

Line 520 kills the sound.

Line 540 checks to see if Grendel hit Beowulf; if so, his score is incremented by one.

Line 550 goes to the subroutine that displays the score.

Line 560 checks the score. If Grendel has slashed Beowulf ten times, the game is over.

Line 570 sets the odds for the game. Grendel gets to take two shots at Beowulf for each of his.

Line 580 branches back to the next complete main program loop.

Line 590 zeroes the odds. Once Grendel has slashed out twice, it's Beowulf's turn.

Lines 600 to 640 move Beowulf horizontally.

Line 650 waits for the trigger button to be pushed. Until it is, line 670 branches back to line 600. We're waiting, Beowulf. . . .

Line 690 puts a light saber, by means of Player/Missile graphics, in Beowulf's hand. Line 700 draws it, as the laser light beams out toward Grendel.

Line 710 makes a laser sound.

Line 720 makes sure the laser/missile is drawn from the bottom to the top of the screen; if at the top, then drop down to 740.

Line 730 keeps us in the draw loop until the missile reaches the top of the screen.

Line 760 checks to see if Beowulf struck Grendel. If Beowulf strikes Grendel ten times, he wins.

Line 800 clears the collision register. That's where we checked to

see if there was a hit in lines 540 and 760. Let's empty it so we get an accurate count.

Lines 810 through 870 print the new score each time one of the players strikes a hit.

Lines 880 through 910 print the lose routine.

Lines 950 through 990 print the win routine.

Lines 1020 through 1090 initialize the players and variables for a new game.

Lines 1100 through 1120 comprise the WAIT subroutine.

Lines 1130 through 1150 comprise the MUSIC subroutine.

Lines 1170 through 1190 hold the musical notes (TONE).

Lines 1200 through 1240 draw the playing field.

Lines 1260 through 1510 initialize the Player/Missile graphics memory locations, and draw both Beowulf and Grendel.

Lines 1520 through 1910 print the instructions.

String and Numeric Variables

A$	Need instructions?
C	Random number used to calculate Grendel's position.
RED	Grendel's horizontal position.
YELLOW	Beowulf's horizontal position.
BEO	Beowulf's score.
GRE	Grendel's score.
MEM	Player/Missile graphics memory location.
ODDS	Tells how many shots Grendel has taken this turn.
DELAY	Plays musical TONE for duration of DELAY.
TONE	Musical notes.
LOOP	Number of notes being played.
PMBASE	Pointer to beginning of Player/Missile memory.
TOP	Memory address 2K below top of RAM.
SABER	Beowulf's missile (light-SABER) Player/Missile memory location.

The Program

```
100 REM BEOWULF VS. GRENDEL GAME
110 PRINT CHR$(125):REM CLEAR SCREEN
120 WAIT=1100:MUSIC=1130
130 LOSE=880:WIN=950:SCORE=820
140 GRAPHICS 2:SETCOLOR 4,8,4
150 REM LOCATE NEXT PRINT STATEMENT
160 POSITION 4,4
170 REM PRINT IN BIG LETTERS
180 ? #6;"B E O W  U L F"
190 GOSUB MUSIC:REM GO PLAY TUNE
200 DIM A$(1)
210 POKE 755,0:REM HIDE CURSOR
220 ? "INSTRUCTIONS (Y/N)":;INPUT A$
230 IF A$<>"N" THEN GOSUB 1520
240 GOTO 1200:REM GO DRAW PLAYFIELD
250 REM BEGIN BATTLE
260 REM GET RANDOM NUMBER
270 LET C=INT(9*RND(1))
280 REM CONVERT TO GRENDEL'S RIGHT OR LEFT MOVE
290 IF C=0 THEN LET POS=10
300 IF C=1 THEN LET POS=-10
310 IF C=2 THEN LET POS=20
320 IF C=3 THEN LET POS=-20
330 IF C=4 THEN LET POS=5
340 IF C=5 THEN LET POS=-25
350 IF C=6 THEN LET POS=15
360 IF C=7 THEN LET POS=-15
370 IF C=8 THEN LET POS=25
380 IF C=9 THEN LET POS=-5
390 REM CALCULATE MONSTER'S MOVE
400 RED=RED+POS
410 IF RED-YELLOW>15 THEN RED=YELLOW
420 IF YELLOW-RED>15 THEN RED=YELLOW
430 IF BEO>GRE+3 THEN RED=YELLOW
440 REM LOCATE MONSTER
450 POKE 53248,RED
460 REM GIVE GRENDEL LIGHT SABER
470 POKE 53252,RED+7
480 REM MONSTER SLASHES OUT
490 FOR MEM=38 TO 83 STEP 3:POKE PMBASE+MEM,3:NEXT MEM
500 FOR MEM=38 TO 83 STEP 3:POKE PMBASE+MEM,0:SOUND 1,MEM-21,6,10:NEXT
MEM
510 FOR DELAY=1 TO 30:NEXT DELAY:REM HOLD SOUND
520 SOUND 1,0,0,0:REM KILL SOUND
530 REM CHECK FOR HIT
540 IF PEEK(53256)=2 THEN GRE=GRE+1
550 GOSUB SCORE
560 IF GRE>10 THEN GOTO LOSE
570 ODDS=ODDS+1:IF ODDS>1 THEN 590
580 GOTO 270
590 ODDS=0
600 REM READ JOYSTICK
610 IF STICK(0)=11 THEN IF YELLOW>90 THEN YELLOW=YELLOW-3
620 IF STICK(0)=7 THEN IF YELLOW<160 THEN YELLOW=YELLOW+3
630 REM MOVE YELLOW (BEOWULF)
640 POKE 53249,YELLOW
650 IF STRIG(0)=0 THEN GOTO 690
660 REM WAIT FOR JOYSTICK TO MOVE
670 GOTO 600
680 REM BEOWULF SLASHES OUT
690 SABER=PMBASE+83:POKE 53253,YELLOW+3
```

```
700 IF SABER>0 THEN POKE SABER,0:POKE SABER-1,0:SABER=SABER-8:POKE SABE
R,12:POKE SABER-1,12
710 SOUND 0,ABS(SABER+20-PMBASE)*0.5,6,14
720 IF SABER<PMBASE+8 THEN GOTO 740
730 GOTO 700:REM MOVE LIGHT BEAM
740 SOUND 0,0,0,0:REM KILL SOUND
750 REM CHECK FOR HIT
760 IF PEEK(53257)=1 THEN BEO=BEO+1
770 IF BEO>10 THEN GOTO WIN
780 GOSUB SCORE
790 REM CLEAR HIT REGISTERS
800 POKE 53278,255
810 GOTO 270:REM MONSTERS TURN AGAIN
820 POKE 84,22
830 PRINT CHR$(125):REM CLEAR SCREEN
840 REM UPDATE SCORE
850 ? "BEOWULF   ";BEO;"  GRENDEL  ";GRE
860 POKE 755,0:REM HIDE CURSOR
870 RETURN
880 SETCOLOR 4,3,1:REM MAKE BORDER BLEED
890 ? "YOU'VE BEEN HACKED TO BITS"
900 ? "BY A BIG, DUMB, MONSTER . . ."
910 POKE 755,0:REM HIDE CURSOR
920 GOSUB MUSIC
930 GOSUB WAIT
940 GOTO 1020
950 SETCOLOR 4,12,10:REM HIGHLIGHT BORDER
960 ? "GOOD SHOW."
970 ? "HE WAS ONLY A MONSTER"
980 ? "HOW ARE YOU WITH DRAGONS ?"
990 POKE 755,0:REM HIDE CURSOR
1000 GOSUB MUSIC
1010 GOSUB WAIT
1020 PRINT :PRINT :PRINT
1030 REM CLEAR FOR NEW GAME
1040 SETCOLOR 4,8,8
1050 POKE 53248,90:POKE 53249,120
1060 POKE 53278,255
1070 BEO=0:GRE=0
1080 REM START OVER
1090 GOTO 270
1100 POKE 764,255
1110 IF PEEK(764)=255 THEN 1110
1120 POKE 764,255:RETURN
1130 FOR LOOP=1 TO 16:READ TONE:SOUND 0,TONE,10,8
1140 FOR DELAY=1 TO 80:NEXT DELAY:SOUND 0,0,10,8
1150 NEXT LOOP:RESTORE :RETURN
1160 REM MUSICAL NOTES STORED AS DATA
1170 DATA 109,109,122,109,92,92
1180 DATA 73,61,65,82,73,65
1190 DATA 109,109,122,109,92,92
1200 GRAPHICS 8:REM DRAW PLAYING FIELD
1210 COLOR 3
1220 PLOT 80,30:DRAWTO 30,149
1230 DRAWTO 289,149:DRAWTO 250,30
1240 DRAWTO 80,30
1250 REM INITIALIZE PLAYER GRAPHICS
1260 TOP=PEEK(106)-8:POKE 54279,TOP
1270 PMBASE=TOP * 256+384
1280 REM CLEAR MEMORY WITH ZEROS
1290 FOR MEM=PMBASE TO PMBASE+384
1300 POKE MEM,0:NEXT MEM
1310 REM DRAW GRENDEL
1320 POKE PMBASE+161,24:POKE PMBASE+162,255
```

```
1330 POKE PMBASE+163,24:POKE PMBASE+164,255
1340 POKE PMBASE+165,255
1350 REM GIVE COLOUR AND POSITION
1360 POKE 704,48:REM PAINT RED
1370 REM DRAW BEOWULF
1380 POKE PMBASE+335,24:POKE PMBASE+336,255
1390 POKE PMBASE+337,24:POKE PMBASE+338,255
1400 POKE PMBASE+339,24:POKE PMBASE+340,255
1410 POKE PMBASE+341,255:POKE PMBASE+342,255
1420 REM GIVE COLOUR AND POSITION
1430 POKE 705,255:REM PAINT YELLOW
1440 POKE 53256,1:REM SET PLAYER SIZE
1450 POKE 53277,3:POKE 559,46
1460 RED=120:YELLOW=120
1470 REM PLACE PLAYERS ON SCREEN
1480 POKE 53248,RED
1490 POKE 53249,YELLOW
1500 REM GO PLAY GAME
1510 GOTO 270
1520 GRAPHICS 2+16
1530 PRINT :PRINT :REM PRINT INSTRUCTIONS
1540 ? "BEOWULF WAS A HEROIC WARRIOR."
1550 PRINT
1560 ? "GRENDEL WAS A LOATHSOME AND HIDEOUSLY"
1570 ? "UGLY CREATURE."
1580 PRINT
1590 ? "SEEMS GRENDEL SKULKED OUT OF THE "
1600 ? "SWAMPS, TORE BIG DOORS OFF HINGES AND"
1610 ? "GRABBED A WARRIOR AND CARTED HIM OFF"
1620 ? "TO HIS LAIR TO DEVOUR."
1630 PRINT
1640 ? "HEROIC AND FEARLESS AS YOU ARE, YOU"
1650 ? "HAVE VOLUNTEERED TO SLAY THE EVIL"
1660 ? "GRENDEL ONE.  "
1670 PRINT
1680 ? "NATURALLY THERE'S A CATCH . . ."
1690 POKE 84,21
1700 ? "  (PRESS ANY KEY TO CONTINUE)":POKE 755,0
1710 GOSUB WAIT
1720 PRINT CHR$(125):REM CLEAR SCREEN
1730 PRINT :PRINT
1740 ? "THE THING IS, THIS IS THE 21ST      "
1750 ? "CENTURY AND GRENDEL HAS A LIGHT SABER"
1760 ? "YOU DO TOO, OF COURSE, BUT YOURS IS "
1770 ? "AN OLDER, SLOWER MODEL. NOT TO WORRY."
1780 ? "GRENDEL IS NOTHING BUT A BIG, DUMB"
1790 ? "MONSTER."
1800 PRINT
1810 ? "GRENDEL IS THE RED PLAYER. YOU ARE "
1820 ? "THE YELLOW ONE."
1830 PRINT
1840 ? "USE THE JOYSTICK TO MANEUVER ACROSS"
1850 ? "THE BATTLEGROUND. HIT THE TRIGGER"
1860 ? "BUTTON TO SLASH OUT WITH THE LIGHT"
1870 ? "SABER.  STAY OUT OF CORNERS."
1880 POKE 84,21
1890 ? "  (HIT ANY KEY TO CONTINUE)":POKE 755,0
1900 GOSUB WAIT
1910 RETURN
```

10 / Helicopter War

Helicopter war pits a Cobra gunship against Vietcong supply lines rumbling their way south. You are pilot and gunner. Fire the mini-gun as the supply trucks, tanks, and jeeps line up in your sights.

Score a hit, and the enemy target disappears. Be watchful. The enemy troops are armed with surface-to-air missiles (SAMs) and are deadly shots. They shoot back. When you take a hit, you'll hear the warning buzzer scream as the huey (helicopter) helplessly crashes and burns. Once five of your airships are shot down, the air battle is over.

To fight another helicopter war, hold the trigger button down until the lead gunship takes off.

For variety, rewrite the display. Change gunship to British, and Vietcong to Argentines. Modify the program to represent the war of the week.

How the Program Works

Lines 110 through 170 blank the screen, and format and print the wait message.

Line 180 branches to the assembly language routine for the helicopter's up and down motion.

Lines 200 through 230 initialize the variables.

Lines 260 through 700 hold the main program loop.

Lines 280 through 330 move the ground vehicles horizontally.

Lines 350 through 520 control the helicopter's up and down and right and left motion.

Lines 550 through 630 locate and fire the ground vehicle's surface-to-air missiles.

Lines 650 through 670 check the collision registers to see if the SAMs hit the helicopter.

Line 680 makes the helicopter rotor's sound.

Lines 730 through 810 fire the helicopter's mini-gun.

Lines 860 through 920 check the collision register, to see if the mini-gun bullets hit one of the ground vehicles.

Lines 970 through 1060 crash the disabled 'copter.

Lines 1070 through 1130 increment the helicopter's score.

Line 1150 paints the 'copter red once it hits ground.

Lines 1170 through 1190 create the exploding-on-impact sound.

Line 1230 checks whether five 'copters have been shot down; if so, that's the end of the game.

Lines 1240 through 1350 initialize the next game, and put the new helicopter in the air.

Lines 1370 through 1500 display the final game score. Holding down the trigger button starts a new game.

Lines 1500 through 1810 draw the helicopter, tank, truck, and jeep.

Lines 1820 through 1920 poke the helicopter up and down assembly language routine into memory.

String and Numeric Variables

THEM	Enemy's score.
US	Helicopter's score.
CRASHED	Number of helicopters shot down.
Y	Altitude of helicopter.
TRUCK	Horizontal position of truck.
TANK	Horizontal position of tank.
JEEP	Horizontal position of jeep.
A	Joystick reading.
B	Joystick trigger reading.

MIS	Height of SAM #1.
MI2	Height of SAM #2.
MI3	Height of SAM #3.
REG	Collision register.
MEM	'Copter's mini-gun bullets drawn in P/M memory.
GROUND	Starting memory location for MIS, MI2, and MI3.
TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.

The Program

```
100 REM HELICOPTER WAR
110 PRINT CHR$(125):REM CLEAR SCREEN
120 GRAPHICS 2+16:REM SET GRAPHICS MODE
130 REM LOCATE CURSOR
140 POSITION 3,4
150 REM PRINT WITH BIG LETTERS
160 ? #6;"PLEASE WAIT"
170 REM LOAD MACHINE LANGUAGE SUBROUTINE
180 GOSUB 1820:GOSUB 1890
190 REM ASSIGN VALUES TO VARIABLES
200 THEM=0:REM THEM IS ENEMY SCORE
210 US=0:REM US IS HELICOPTER SCORE
220 CRASHED=0:REM HELICOPTERS SHOT DOWN
230 Y=100:REM HELICOPTER ALTITUDE
240 REM GO DRAW PLAYERS
250 GOTO 1560
260 REM MAIN PROGRAM LOOP
270 REM INCREMENT GROUND VEHICLES POSITION
280 TRUCK=TRUCK+2:IF TRUCK>239 THEN TRUCK=20
290 TANK=TANK+1.5:IF TANK>239 THEN TANK=20
300 JEEP=JEEP+4:IF JEEP>239 THEN JEEP=20
310 POKE 53249,TRUCK:REM MOVE TRUCK
320 POKE 53250,TANK:REM MOVE TANK
330 POKE 53251,JEEP:REM MOVE JEEP
340 REM ASSIGN VARIABLE TO STICK
350 LET A=STICK(0):LET B=STRIG(0)
360 REM CHECK MINI-GUN TRIGGER
370 IF B=0 THEN GOTO 730
380 REM DRIFT COPTER TO LEFT OF SCREEN
390 COPTER=COPTER-1:POKE 53248,COPTER
400 REM MOVE COPTER LEFT
410 IF A=11 THEN COPTER=COPTER-5:POKE 53248,COPTER
420 REM MOVE COPTER RIGHT
430 IF A=7 THEN COPTER=COPTER+7:POKE 53248,COPTER
440 REM KEEP COPTER ON SCREEN
450 IF COPTER<50 THEN COPTER=200
460 IF COPTER>200 THEN COPTER=50
470 REM READ JOYSTICK TO DIVE
```

```
480 IF A=13 THEN D=USR(DOWN,PMBASE+511+Y):Y=Y+1
490 REM READ JOYSTICK TO CLIMB
500 IF A=14 THEN D=USR(UP,PMBASE+511+Y):Y=Y-1
510 REM KEEP FROM FLYING OFF TOP
520 IF Y=5 THEN D=USR(DOWN,PMBASE+511+Y):Y=Y+1
530 REM CHECK DISTANCE TO COPTER. IF IN RANGE,
540 REM THEN FIRE SURFACE-TO-AIR MISSILE
550 IF ABS(COPTER-TRUCK)<2 THEN POKE 53253,TRUCK-5:MIS=GROUND+90
560 IF MIS>0 THEN POKE MIS,0:POKE MIS-1,0:MIS=MIS-5:POKE MIS,8:POKE MIS
-1,8
570 IF ABS(COPTER-TANK)<2 THEN POKE 53254,TANK-5:MI2=GROUND+90
580 IF MI2>0 THEN POKE MI2,0:POKE MI2-1,0:MI2=MI2-8:POKE MI2,32:POKE MI
2-1,32
590 IF ABS(COPTER-JEEP)<2 THEN POKE 53255,JEEP-5:MI3=GROUND+90
600 IF MI3>0 THEN POKE MI3,0:POKE MI3-1,0:MI3=MI3-8:POKE MI3,128:POKE M
I3-1,128
610 IF MI3<GROUND+10 THEN MI3=0
620 IF MI2<GROUND+10 THEN MI2=0
630 IF MIS<GROUND+10 THEN MIS=0
640 REM READ COLLISION REGISTERS
650 IF PEEK(53257)=1 THEN 970
660 IF PEEK(53258)=1 THEN 970
670 IF PEEK(53259)=1 THEN 970
680 SOUND 2,35,6,15:SOUND 2,0,0,0
690 REM LOOP BACK TO FOR NEXT MOVE
700 GOTO 280
710 REM MINI-GUN FIRING SEQUENCE
720 REM GIVE COPTER MINI-GUN
730 POKE 53252,COPTER+5
740 REM CONTINUE WITH COPTER SOUND
750 SOUND 2,35,6,15:SOUND 2,0,0,0
760 REM LOCATE MISSILE IN MEMORY
770 GROUND=PMBASE+384
780 REM DRAW MINI-GUN BULLETS
790 FOR MEM=Y TO 104 STEP 3.5:POKE GROUND+MEM,3:NEXT MEM
800 REM UN-DRAW BULLETS, MAKE SOUND
810 FOR MEM=Y TO 104 STEP 3.5:POKE GROUND+MEM,0:SOUND 1,199,6,15:NEXT M
EM
820 SOUND 1,0,0,0:REM KILL SOUND
830 REM CONTINUE COPTER SOUND
840 SOUND 2,35,6,15:SOUND 2,0,0,0
850 REM READ COLLISION REGISTERS
860 LET REG=PEEK(53256)
870 REM IF HIT, ERASE GROUND VEHICLE
880 IF REG=3 THEN TRUCK=16:GOTO 1070
890 IF REG=5 THEN TANK=16:GOTO 1070
900 IF REG=9 THEN JEEP=16:GOTO 1070
910 REM IF MISS THEN IGNORE
920 IF REG>9 THEN POKE 53278,255
930 SOUND 2,35,6,15:SOUND 2,0,0,0
940 REM BACK TO MAIN PROGRAM LOOP
950 GOTO 300
960 REM SCORE ROUTINE
970 IF Y <102 THEN D = USR(DOWN,PMBASE+511+Y):Y = Y+1:SOUND 0,50,12,6:G
OTO 970
980 POKE 53278,255:SOUND 0,0,0,0
990 CRASHED=CRASHED+1:REM ONE MORE COPTER DOWN
1000 PRINT #6;CHR$(125):REM ERASE SCORE
1010 THEM=THEM+25:REM INCREMENT SCORE
1020 REM LOCATE CURSOR
1030 POSITION 2,0
1040 PRINT #6;"VIETCONG  ";THEM
1050 REM GOTO EXPLOSION
1060 GOTO 1150
```

```
1070 US=US+5:REM INCREMENT SCORE
1080 PRINT #6;CHR$(125):REM ERASE SCORE
1090 POSITION 2,0
1100 PRINT #6;"GUNSHIP  ";US
1110 REM CLEAR COLLISION REGISTER
1120 POKE 53278,255
1130 GOTO 300
1140 REM EXPLODE COPTER ON IMPACT
1150 POKE 704,48
1160 REM CREATE EXPLOSION SOUND
1170 FOR TONE=15 TO 0 STEP -1:SOUND 0,25,16,TONE
1180 REM HOLD SOUND FOR INSTANT
1190 FOR DELAY=1 TO 20:NEXT DELAY:NEXT TONE
1200 REM SHUT OFF SOUND
1210 SOUND 1,0,0,0:SOUND 0,0,0,0
1220 REM GAME OVER WHEN FIVE SHOT DOWN
1230 IF CRASHED=5 THEN 1370
1240 POKE 53278,255:REM CLEAR HIT REGISTER
1250 COPTER=120:REM MOVE COPTERR
1260 POKE 53248,COPTER
1270 LET A=STICK(0):REM ASSIGN JOYSTICK
1280 POKE 704,200:REM PAINT NEW COPTER GREEN
1290 REM FLY  NEW COPTER INTO SKY
1300 IF Y>20 THEN D=USR(UP,PMBASE+511+Y):Y=Y-1
1310 REM WHEN ALTITUDE HIGH ENOUGH BEGIN GAME
1320 IF Y<=20 THEN GOTO 280
1330 REM MOVE GROUND VEHICLES
1340 TRUCK=20:TANK=20:JEEP=20
1350 GOTO 1300
1360 REM DISPLAY SCORE
1370 POSITION 2,0
1380 REM DISPLAY SCORE IN BIG LETTERS
1390 PRINT #6;"VIETCONG  ";THEM
1400 FOR DELAY=1 TO 200:NEXT DELAY
1410 PRINT #6;CHR$(125):REM ERASE SCORE
1420 POSITION 2,0
1430 REM DISPLAY SCORE
1440 PRINT #6;"GUNSHIP  ";US
1450 REM DELAY LOOP
1460 FOR DELAY=1 TO 200:NEXT DELAY
1470 REM HOLD DOWN TRIGGER FOR NEXT GAME
1480 IF STRIG(0)=0 THEN 1520
1490 REM LOOP UNTIL NEXT GAME
1500 GOTO 1370
1510 REM BLANK SCREEN THEN START NEW GAME
1520 THEM=0:US=0:CRASHED=0:REM START OVER
1530 PRINT #6;CHR$(125):REM ERASE SCORE
1540 GOTO 1240
1550 REM DRAW HELICOPTER AND TANKS
1560 TOP=PEEK(106)-8:POKE 54279,TOP
1570 PMBASE=256*TOP:GROUND=PMBASE+384
1580 REM CLEAR GRAPHICS MEMORY W/ZEROS
1590 FOR MEM=PMBASE+384 TO PMBASE+1024:POKE MEM,0:NEXT MEM
1600 REM DRAW HELICOPTER
1610 POKE PMBASE+612,255:POKE PMBASE+613,40:POKE PMBASE+614,127
1620 POKE PMBASE+615,122:POKE PMBASE+616,132
1630 REM DRAW BIG TRUCK
1640 POKE PMBASE+740,248:POKE PMBASE+741,254
1650 POKE PMBASE+742,255:POKE PMBASE+743,255:POKE PMBASE+744,82
1660 REM DRAW TANK
1670 POKE PMBASE+860,112:POKE PMBASE+861,127
1680 POKE PMBASE+862,112:POKE PMBASE+863,255
1690 POKE PMBASE+864,170:POKE PMBASE+865,84
1700 REM DRAW JEEP
```

```
1710 POKE PMBASE+990,248:POKE PMBASE+991,254
1720 POKE PMBASE+992,255:POKE PMBASE+993,82
1730 REM GET MISSILE ADDRESS
1740 GROUND=PMBASE+384
1750 REM PAINT GROUND VEHICLES
1760 POKE 705,85:POKE 706,130:POKE 707,150
1770 REM TURN ON GRAPHICS
1780 POKE 53277,3:POKE 559,46
1790 REM ERASE WAIT MESSAGE
1800 POSITION 0,0;PRINT #6:CHR$(125)
1810 GOTO 1250
1820 DIM UPCODE$(22):UP=ADR(UPCODE$)
1830 FOR I=UP TO UP+20
1840 READ BYTE:POKE I,BYTE
1850 NEXT I:RETURN
1860 DATA 104,104,133,204,104,133,203
1870 DATA 160,1,177,203,136,145,203
1880 DATA 200,200,192,7,208,245,96
1890 DIM DOWNCODE$(22):DOWN=ADR(DOWNCODE$)
1900 FOR I=DOWN TO DOWN+20
1910 READ BYTE:POKE I,BYTE
1920 NEXT I:RESTORE :RETURN
1930 DATA 104,104,133,204,104,133,203
1940 DATA 160,6,177,203,200,145,203
1950 DATA 136,136,192,255,208,245,96
```

11 / Jet Jockey

Dateline: Blue skies over the Rhine River valley. The world is at war. You are strapped into the cockpit of a Tomcat F-14 fighter plane. Armed with air-to-air rockets, you monitor the radar screen, watching for the blips of attacking Soviet planes.

When one appears, jockey your plane to get a clear shot at Ivan. Each attack comes at a different speed. If he gets within range, he'll fire a missile at you. Ivan is well trained. If he fires, you have a 50-50 chance of going down.

Fly your plane with the joystick. Fire the rockets to defend yourself. One after the other, swarms of Red warplanes will come at you. Keep firing until your squadron is shot down. The score will then flash on the screen to tell how good a shot you are. To play again, hold down the trigger button.

How the Program Works

Line 110 blanks the screen

Line 120 branches to the Player/Missile graphics drawing routine.

Line 150 begins the main program loop. The ATARI Home Computer supplies a random number between 0 and 8. This number determines the speed of the enemy plane, painted red, as it flies across the screen. Each time the plane flies at a different speed. The bigger the number in "X * RND", the faster the plane can fly. Plug in a 20 or 30 instead of the 8, and see what happens.

Line 180 moves the plane with a loop. Line 180 gets a number starting with 25, adds speed to it, and places the red plane at that horizontal position. The next time through the loop, the ATARI Home Computer adds the number stored in SPEED to the previous number, and advances the red plane to that number.

Lines 220 and 230 read the joystick and move the white plane.

Line 250 reads the joystick trigger; if pressed, it fires a missile.

Line 270 creates the radar beep-beeping sound.

Line 300 draws the good guy's missile.

Line 320 stops the good guy's missile once it's just past the top of the screen.

Line 340 compares the distance from the red plane to the white plane. If in range, the red plane fires its missile.

Line 350 draws the red missile.

Line 370 stops the red missile once it's below the bottom of the screen.

Lines 390 and 410 create the missile's sound effects.

Lines 440 and 460 check the collision registers to see if either plane has been hit.

Lines 520 through 550 create the explosion sound effect.

Line 560 clears the collision register if either plane has taken a hit.

Lines 600 through 660 format the screen and print the title message. Lines 670 through 720 sound the alarm.

Lines 770 through 1030 draw both planes in Player/Missile memory.

String and Numeric Variables

TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.
MEM	Temporary address for drawing Player/Missile graphics.
WHITE	Horizontal position of good guy's jet.
MIS	Memory location of white missile on way to target.
ROCKET	Memory location of red missile on way to target.

BYTE	Memory loction where white plane is drawn.
AUDIO	Sound effects tone.
SPEED	Horizontal speed of red plane.
MOVE	Horizontal position of red plane.
DELAY	Duration tone plays.

The Program

```
100 REM JET JOCKEY GAME
110 PRINT CHR$(125):REM CLEAR SCREEN
120 GOSUB 600:REM GO DRAW PLANES
130 REM BEGIN GAME
140 REM VARY SPEED OF ENEMY PLANE
150 LET SPEED=INT(8*RND(1))
160 IF SPEED<2 THEN 150
170 REM MAIN PROGRAM LOOP
180 FOR MOVE=25 TO 230 STEP SPEED
190 REM MOVE RED PLANE
200 POKE 53248,MOVE
210 REM MOVE WHITE PLANE
220 IF STICK(0)=11 THEN IF WHITE>50 THEN WHITE=WHITE-8:POKE 53249,WHITE
230 IF STICK(0)=7 THEN IF WHITE<198 THEN WHITE=WHITE+8:POKE 53249,WHITE
240 REM FIRE MISSILE
250 IF STRIG(0)=0 THEN MIS=ADD+90:POKE 53253,WHITE
260 REM CREATE RADAR SOUND
270 AUDIO=AUDIO+1:IF AUDIO>6 THEN AUDIO=1
280 SOUND 0,21/AUDIO,10,10
290 REM FIRE OUR MISSILE
300 IF MIS>ADD THEN POKE MIS,0:POKE MIS-1,0:MIS=MIS-5:POKE MIS,8:POKE M
IS-1,8
310 REM ERASE OLD MISSILE
320 IF MIS<=ADD THEN MIS=0
330 REM FIRE RED ROCKET
340 IF ABS(WHITE-MOVE)<5 THEN POKE 53252,MOVE+3:ROCKET=ADD+30
350 IF ROCKET>0 THEN POKE ROCKET,0:POKE ROCKET-1,0:ROCKET=ROCKET+5:POKE
 ROCKET,2:POKE ROCKET-1,2
360 REM ERASE OLD RED ROCKET
370 IF ROCKET=ADD+125 THEN ROCKET=0
380 REM RED MISSILE SOUND
390 SOUND 1,ABS(MIS),10,10
400 REM WHITE ROCKET SOUND
410 SOUND 2,ABS(ROCKET),10,10
420 REM SEE IF RED PLANE HIT
430 REM IF HIT ERASE, DO EXPLODE
440 IF PEEK(53257)=1 THEN POKE 53253,1:POKE 53248,20:GOTO 520
450 REM SEE IF WHITE PLANE HIT
460 REM IF HIT ERASE, DO EXPLODE
470 IF PEEK(53256)=2 THEN POKE 53252,1:POKE 53249,20:GOTO 520
480 NEXT MOVE
490 REM NEXT RED PLANE ATTACKS
500 GOTO 150
510 REM BANG ROUTINE
520 FOR TONE=15 TO 0 STEP -1
530 SOUND 0,25,16,TONE
540 FOR DELAY=1 TO 7:NEXT DELAY
550 NEXT TONE
```

```
560 POKE 53278,255
570 POP
580 GOTO 150
590 REM DRAW BOTH PLANES
600 REM COLOR PLAY FIELD
610 GRAPHICS 2:REM CHOOSE PLAYFIELD
620 SETCOLOR 4,8,4
630 SETCOLOR 2,8,4
640 REM POSITION CURSOR
650 POSITION 5,2
660 PRINT #6;"JET JOCKEY"
670 FOR LOOP=1 TO 19:REM BEGIN SOUND LOOP
680 FOR TONE=20 TO 40
690 SOUND 2,TONE,10,8
700 NEXT TONE
710 SOUND 2,0,0,0
720 NEXT LOOP
730 PRINT #6;CHR$(125)
740 POSITION 3,3
750 PRINT #6;"PLEASE WAIT"
760 REM INITIALIZE PLAYER MISSILE GRAPHICS
770 TOP=PEEK(106)-8:POKE 54279,TOP
780 PMBASE=TOP * 256
790 REM CLEAR P/M MEMORY WITH ZEROS
800 FOR MEM=PMBASE+384 TO PMBASE+767
810 POKE MEM,0:NEXT MEM
820 REM DRAW AMERICAN
830 FOR MEM=PMBASE+730 TO PMBASE+735
840 READ BYTE:POKE MEM,BYTE:NEXT MEM
850 DATA 16,254,124,16,16,56
860 REM DRAW ENEMY
870 FOR MEM=PMBASE+540 TO PMBASE+546
880 READ BYTE:POKE MEM,BYTE:NEXT MEM
890 DATA 4,12,140,255,140,12,4
900 REM COLOR ENEMY RED, U.S. WHITE
910 POKE 704,50:POKE 705,175
920 REM SET PLAYER SIZE
930 POKE 53257,1:POKE 53256,0
940 POKE 559,46
950 POKE 53277,3:REM SWITCH ON GRAPHICS
960 REM SET-UP VARIABLES
970 MIS=0
980 WHITE=120:REM U.S. HORIZ POSITION
990 POKE 53249,WHITE
1000 POKE 53253,WHITE
1010 ADD=PMBASE+384:REM MISSILE MEMORY LOCATION
1020 PRINT #6;CHR$(125):REM CLEAR SCREEN
1030 RETURN
```

12 / Bridge Buster

Bridge Buster is a war game. You are behind enemy lines with one hundred pounds of C-4 plastic explosives. Your mission: blow up the bad guys' bridge. There are a number of complications.

First, you have to use at least five pounds, and no more than the one hundred pounds of the C-4 you infiltrated with.

Second, even though you've been specially trained in using explosives, you don't know how many pounds it will take to drop this span into the gulch. You'll have to guess.

If you use too much, you'll blow away yourself and your buddies in addition to the bridge. These days there's little glory in being a posthumous hero. On the other hand, if you don't place a big enough charge, there will be a bang, but little else. If you have any explosives left, you can try again. But don't react out of frustration and overdo it.

To help you along, there will be clues, such as "Not quite enough", "Way off", as well as a tally of just how many pounds of explosives you've expended so far. By the way, each bridge is different. No two ever require the same charge. That's because a random number determines the target figure.

Good luck. Should you or your commando force be killed, captured, or identified, we'll be sure to read about it in the papers.

How the Program Works

Lines 100 through 130 initialize the variables.

Lines 270 through 380 draw the little red truck.

Line 430 reads the console buttons; the ATARI Home Computer waits for the START button to be pushed.

Lines 450 through 530 move the truck across the bridge.

Line 550 chooses how many pounds of explosives are needed to blow the bridge.

Line 600 keeps track of how many pounds have been exploded.

Line 620 ends the program if all explosives are used up.

Line 630 tells how many pounds are left.

Line 650 asks for your estimate of pounds needed to bring down the bridge.

Lines 700 and 720 make sure at least five pounds and no more than one hundred are used. Otherwise, you have to start over.

Lines 800 through 840 match how much you estimated against how many pounds were actually needed.

Lines 860 and 890 delay messages display on the screen.

Line 900 blanks the text window.

Line 990 creates the explosion sound effect.

Lines 1010 through 1180 draw the high-resolution bridge.

String and Numeric Variables

MESSAGE$	Warns if over or under explosives limit.
TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.
TRUCK	Horizontal position of red truck.
DRAW	P/M memory where truck is drawn.
CHARGE	Your estimate of pounds needed.
LEFT	Pounds explosives left.
USED	Pounds used up.
NEED	Pounds necessary to blow bridge.

TONE	Sound effects tone.
DELAY	Duration tone is played.

The Program

```
100 REM BRIDGE BUSTER
110 DIM MESSAGE$(34)
120 LET MESSAGE$="AT LEAST 5 POUNDS NO MORE THAN 100"
130 TRUCK=50:REM TRUCK HORIZ. POSITION
140 REM CHOOSE GRAPHICS AND PAINT
150 GRAPHICS 2+16:SETCOLOR 4,8,4
160 REM FORMAT GAME TITLE
170 POSITION 4,3
180 REM PRINT MESSAGE, HIDE CURSOR
190 PRINT #6;"BRIDGE BUSTER":POKE 755,0
200 REM GO MAKE EXPLOSION SOUND
210 GOSUB 980
220 REM DELAY MESSAGE ON SCREEN
230 FOR DELAY=1 TO 230:NEXT DELAY
240 REM GO DRAW BRIDGE
250 GOSUB 1010
260 REM INITIALIZE GRAPHICS
270 TOP=PEEK(106)-8:POKE 54279,TOP
280 PMBASE=256*TOP
290 REM CLEAR MEMORY WITH ZEROS
300 FOR DRAW=PMBASE+512 TO PMBASE+639
310 POKE DRAW,0:NEXT DRAW
320 REM DRAW TRUCK
330 POKE PMBASE+580,248:POKE PMBASE+581,254
340 POKE PMBASE+582,255:POKE PMBASE+583,82
350 REM TURN ON GRAPHICS
360 POKE 53277,3:POKE 559,46
370 POKE 53248,50:REM TRUCK POSITION
380 POKE 704,48:REM COLOR TRUCK RED
390 REM ZERO OUT VARIABLES
400 CHARGE=0:LEFT=0:USED=0:NEED=0
410 PRINT "PRESS YELLOW START BUTTON":POKE 755,0
420 REM READ CONSOLE "START" KEY
430 IF PEEK(53279)=6 THEN 550
440 REM INCREMENT TRUCK MOVE VARIABLE
450 TRUCK=TRUCK+1
460 REM KEEP TRUCK ON SCREEN
470 IF TRUCK>210 THEN TRUCK=50
480 REM MAKE MOTOR SOUND
490 SOUND 1,100,1,10:SOUND 1,0,0,0
500 REM DRAW TRUCK NEW POSITION
510 POKE 53248,TRUCK
520 REM DO LOOP TO KEEP TRUCK MOVING
530 GOTO 430
540 REM RANDOM CHOICE OF DYNAMITE NEEDED
550 LET NEED=INT(10*RND(1)):NEED=NEED*NEED:IF NEED<5 THEN NEED=5
560 PRINT CHR$(125):REM CLEAR SCREEN
570 REM KILL SOUND
580 SOUND 1,0,0,0
590 REM CALCULATE POUNDS LEFT
600 LEFT=100-USED
610 REM IF ALL GONE GAME IS OVER
620 IF LEFT<1 THEN PRINT "SORRY, ";LEFT;" POUNDS LEFT":GOTO 890
```

```
630 PRINT LEFT; POUNDS C-4 REMAIN IN YOUR KIT BAG"
640 REM ENTER POUNDS TO BE USED
650 PRINT "HOW BIG OF A CHARGE";:POKE 755,0:PRINT;:INPUT CHARGE
660 REM USED EQUALS POUNDS EXPENDED
670 REM TALLY TOTAL POUNDS USED THIS BRIDGE
680 USED=CHARGE+USED
690 REM IF OVER LIMIT REPRIMAND
700 IF CHARGE>100 THEN PRINT CHR$(125);MESSAGE$:GOTO 890
710 REM IF NOT ENOUGH THEN REPRIMAND
720 IF CHARGE<5 THEN PRINT CHR$(125);MESSAGE$:GOTO 890
730 REM IF NOT ENOUGH LEFT
740 GOSUB 980:REM GOTO EXPLOSION
750 PRINT CHR$(125):REM CLEAR SCREEN
760 REM SEE HOW CLOSE TO DOWNING BRIDGE
770 IF CHARGE>NEED THEN 820
780 IF CHARGE<NEED THEN 790
790 PRINT CHR$(125):REM CLEAR SCREEN
800 IF NEED-CHARGE>20 THEN PRINT "WAY OFF.  GIVE HER MORE POWDER!":GOTO
 850
810 IF NEED-CHARGE>5 THEN PRINT "NOT QUITE ENOUGH . . .":GOTO 850
820 IF NEED=CHARGE THEN PRINT "KERBLAMMMM !!!  RIGHT ON THE MONEY !!":G
OTO 890
830 IF CHARGE-NEED<10 THEN PRINT "BRIDGE BLOWN AWAY!  GOOD JOB TROOPIE!
":GOTO 890
840 IF CHARGE-NEED>10 THEN PRINT "BLAMMO!  BLEW AWAY YOU AND THE BRIDGE
":GOTO 890
850 REM DELAY MESSAGE ON SCREEN
860 FOR DELAY=1 TO 400:NEXT DELAY
870 GOTO 560
880 REM LOOP DELAYS MESSAGE DISPLAY
890 FOR DELAY=1 TO 400:NEXT DELAY
900 PRINT CHR$(125):REM CLEAR SCREEN
910 REM ZERO VARIABLES
920 CHARGE=0:NEED=0:USED=0:LEFT=0
930 PRINT "HOLD DOWN START TO GET NEW BRIDGE"
940 REM READ CONSOLE SWITCHES
950 IF PEEK(53279)=6 THEN 550
960 REM LOOP TILL SWITCH PRESSED
970 GOTO 950
980 REM EXPLOSION SUBROUTINE
990 FOR TONE=15 TO 0 STEP -1:SOUND 0,25,16,TONE:FOR DELAY=1 TO 15:NEXT
DELAY:NEXT TONE
1000 RETURN
1010 GRAPHICS 8:COLOR 1:REM DRAW BRIDGE
1020 PLOT 80,124:DRAWTO 224,124:DRAWTO 212,112
1030 DRAWTO 136,96:DRAWTO 88,112:DRAWTO 80,124
1040 PLOT 36,159:DRAWTO 76,132
1050 PLOT 108,92:DRAWTO 248,92:DRAWTO 240,84
1060 DRAWTO 172,64:DRAWTO 112,84:DRAWTO 108,92
1070 PLOT 96,124:DRAWTO 88,132:DRAWTO 76,132
1080 DRAWTO 84,159:DRAWTO 96,159:DRAWTO 88,132
1090 PLOT 110,124:DRAWTO 115,134:DRAWTO 96,159
1100 PLOT 148,124:DRAWTO 155,140:DRAWTO 146,150
1110 PLOT 134,124:DRAWTO 130,128:DRAWTO 138,128:DRAWTO 142,124
1120 PLOT 130,128:DRAWTO 134,150:DRAWTO 146,150:DRAWTO 138,128
1130 PLOT 224,124:DRAWTO 224,124:DRAWTO 220,128
1140 DRAWTO 212,128:DRAWTO 218,124
1150 PLOT 212,128:DRAWTO 216,140:DRAWTO 228,140
1160 DRAWTO 220,128:PLOT 0,96:DRAWTO 319,96
1170 PLOT 0,118:DRAWTO 319,118
1180 RETURN
```

13 / .44 Magnum Russian Roulette

Russian Roulette is a deadly game of chance where one holds a revolver against the temple and pulls the trigger. There are five empty chambers and one loaded. It's the ultimate test of luck. This version is electronic, and somewhat safer. But, to spice things up, we'll pretend we're using a .44 Magnum, the world's most powerful handgun.

First, one bullet is loaded into one of the six empty chambers, and the cylinder is spun. It's spun at every turn. So, if five brave players have gone before you and heard click when they pulled the trigger, it doesn't necessarily mean you'll be the one to die. You may go on the first shot, or you may hang on until the seventeenth or eighteenth turn before finally making the transition to the next world.

To play, RUN the program. You'll see:

```
TURN
1 ?
```

At this point, player number one is up. Press any key to simulate pulling the trigger. If the first player's luck held, number two will come up on the screen, and the next poor soul will have to take his chance at cheating computer death. After each successful turn, the next number will come up, signalling the next turn.

How the Program Works

Lines 120 and 130 tell where the subroutines are located. BULLET loads the gun. BLAM fires the shot.

Line 140 blanks the screen.

Line 210 opens the keyboard for a read. This done, it won't be necessary to hit RETURN; just tap any key to simulate pulling the trigger.

Line 220 turns off the cursor for a neater display.

Line 260 counts how many turns have been taken.

Line 290 spins the loaded cylinder.

Line 310 waits for someone to press a key.

Line 330 checks to see if the variable stored in SPIN is the same as stored in BULLET; if so, the gun fires. If not, line 350 sends us on to another turn.

Line 370 loads the six-cylinder revolver with one bullet, between 0 and 5 BULLET is a random number.

Lines 400 and 410 format and print the lose statement.

Lines 430 through 460 create a gunshot.

Line 490 blanks the screen, and line 510 starts a new game.

String and Numeric Variables

BULLET	Tells where subroutine is that loads the .44 Magnum.
BLAM	Tells location of gunshot subroutine.
SPIN	Random number that simulates spinning loaded cylinder.
TURN	Waits for any key to be pressed.
SHOT	Decrementing pitch used in gunshot loop.
DELAY	Duration sound of shot is held.

The Program

```
100 REM .44 MAGNUM GAME
110 REM DEFINE SUBROUTINE LOCATIONS
120 BULLET=370
130 BLAM=400
```

```
140 PRINT CHR$(125):REM CLEAR SCREEN
150 REM GO LOAD GUN
160 GOSUB BULLET
170 REM LOCATE NEXT MESSAGE ON SCREEN
180 POSITION 15,5:PRINT "TURN"
190 PRINT :PRINT
200 REM OPEN KEYBOARD FOR READ
210 OPEN #1,4,0,"K:"
220 POKE 755,0:REM HIDE CURSOR
230 REM MAIN PROGRAM LOOP
240 REM HIT RETURN TO PULL TRIGGER
250 REM KEEP TRACK OF WHOSE TURN
260 NUMBER=NUMBER+1
270 POKE 85,12:PRINT NUMBER
280 REM SPIN CYLINDER
290 SPIN=INT(5*RND(1))
300 REM PULL THE TRIGGER
310 GET #1,TURN
320 REM CHECK FOR BAD LUCK
330 IF SPIN=BULLET THEN GOTO BLAM
340 REM NEXT TURN
350 GOTO 240
360 REM LOAD ONE CYLINDER
370 BULLET=INT(5*RND(1))
380 RETURN
390 REM GUNSHOT ROUTINE
400 POSITION 5,21:REM PRINT ON BOTTOM SCREEN
410 PRINT "BETTER LUCK NEXT TIME"
420 REM CREATE SOUND OF SHOT
430 FOR SHOT=15 TO 0 STEP -1
440 SOUND 0,35,16,SHOT
450 FOR DELAY=1 TO 15
460 NEXT DELAY:NEXT SHOT
470 REM END SOUND
480 SOUND 0,0,0,0
490 PRINT CHR$(125):REM CLEAR SCREEN
500 REM PLAY ANOTHER GAME
510 RUN
```

14 / Oracle at Delphi

For thousands of years, people have been possessed with a desire to peek into the future. Even in these days of high technology, the popularity of psychics and science fiction continues. While the Oracle at Delphi is a game, it also can be taken seriously.

At university and other accredited paranormal research centers, the standard test for extra sensory perception, ESP, requires the subject to guess which card is next going to be drawn from a deck. The Oracle at Delphi, however, is no ordinary poker deck. Instead of Kings, Queens, Jacks, and Aces, there are colored stars, pyramids, and boxes.

One day, a research assistant noticed that some of the subjects who hadn't been considered psychic actually were. They had been predicting the order the cards would be drawn, such as, "star, box, box, pyramid." The cards were coming up circle, star, box, box, pyramid. They were predicting the exact sequence; the only mistake was when that sequence actually began. In playing the oracle game, you might guess the order each of the symbols will appear on the TV. If your answers are wrong, check to be sure your pattern matches the machine's, but with one or two symbols out of synchrony.

Similarly, the Oracle can be used as an ESP test. Have the subject become familiar with the symbols and sayings, and ask him or her to predict which one will appear next.

This game can be used also to channel our intuition. Most of us have ESP, but we haven't used it in so long that the ability is hidden away, rusting from disuse. Naturally, when our ESP does break through, it's inconsistent and inaccurate—and hard to trust. However, by using this game as an ESP test, or as a fortune telling session, our intuition "muscles" are exercised. With practice, intuition

becomes consistent and reliable. While we may no longer need our intuition to sense the presence of a prowling saber-toothed tiger, we might instead focus the energy on how to best approach a grumbly spouse or make a business decision. For more insight on creatively using your intuition, see Ralph Blum's *Book of Runes* (New York: St. Martin's Press, 1983).

A final consideration: can brain power affect microprocessor decision making? To find out, concentrate. Envision which symbol you want to come up next. Will it to do so. Hold your hand on the keyboard and let the Oracle cycle through a few turns. Of those few turns, how many times did your will win out over electronics? The answer may surprise you.

While this is a long program, all of it need not be entered in order to use it. That's because each set of symbol and text is like a color slide. Type in as many as you want until you get tired of typing. Fill in the rest at a later date. The program will run fine without them. Or, if you want to add even more symbols and sayings than the program provides, add them. In order to do this, increase the random number located in line 310, from 30 to however many slides there are. Also, renumber program lines from 2710 to 2830 to make room for the new slideshow.

How the Program Works

Line 110 selects Graphics Mode 2 so we can print with large letters.

Lines 120 through 190 display the title.

Lines 210 through 260 play the game's theme song.

Lines 340 through 440 draw each of the symbols.

Lines 460 through 510 branch to the various symbols and sayings.

Lines 530 through 610 represent how each of the symbols and sayings are displayed. Lines 550 and 560 draw the symbol; while 570 through 590 print the saying. Line 610 sends us to the subroutine where the ATARI Home Computer waits for any key to be pressed.

Lines 2800 through 2830 comprise the wait routine.

String and Numeric Variables

A$	Want instructions?
TOP	Memory address 2K below top of RAM.
P	Pointer to beginning of Player/Missile memory.
C	Get a random number to choose which symbol and saying to display.

The Program

```
100 REM ORACLE GAME
110 GRAPHICS 2:REM CHOOSE GRAPHICS
120 SETCOLOR 4,7,0:REM COLOR BACKGROUND
130 SETCOLOR 2,7,0:REM COLOR TEXT WIND.
140 POKE 85,7:REM SET HORIZ. MARGIN
150 REM PRINT IN BIG LETTERS
160 PRINT #6;"THE"
170 REM LOCATE CURSOR
180 POSITION 4,4
190 PRINT #6;"O R A C L E":REM SKIP SPACES
200 REM BETWEEN EACH LETTER IN ORACLE
210 FOR LOOP=1 TO 16:READ TONE:SOUND 0,TONE,10,8
220 FOR DELAY=1 TO 200:NEXT DELAY:SOUND 0,0,10,8
230 FOR DELAY=1 TO 10:NEXT DELAY:NEXT LOOP
240 REM DATA FOR MUSICAL NOTES
250 DATA 145,82,73,97,92,109,122,54,48
260 DATA 61,129,65,32,41,46,36
270 DIM A$(1):POKE 755,0
280 PRINT "INSTRUCTIONS (Y/N)";:INPUT A$
290 IF A$<>"N" THEN 2510
300 REM GET RANDOM NUMBER
310 LET C=INT(30*RND(1))+1
320 GRAPHICS 2:REM GRAPHICS AND TEXT
330 REM INITIALIZE GRAPHICS
340 TOP=PEEK(106)-8:POKE 54279,TOP
350 P=256*TOP:POKE 559,46:POKE 53277,3
360 REM CLEAR MEMORY WITH ZEROS
370 FOR DRAW=P+560 TO P+570
380 POKE DRAW,0:NEXT DRAW
390 REM PICK RANDOM COLOR FOR SYMBOL
400 POKE 704,INT(15*RND(1))*16+8
410 REM SET HORIZ. POSITION
420 POKE 53248,120
430 REM SET SYMBOL SIZE
440 POKE 53256,1
450 REM ROUTE TO APPROPRIATE MESSAGE
460 ON C GOTO 520,620,690,760,820
470 ON C-5 GOTO 890,950,1020,1070,1150
480 ON C-10 GOTO 1190,1260,1290,1370,1420
490 ON C-15 GOTO 1490,1560,1640,1710,1760
500 ON C-20 GOTO 2780,1860,1940,2000,2070
510 ON C-25 GOTO 2130,2180,2230,2310,2390
520 POKE 85,7:REM CENTER TEXT
530 PRINT #6;"ARIES"
```

```
540 REM DRAW SYMBOL
550 POKE P+560,238:POKE P+561,170:POKE P+562,170:POKE P+563,16
560 POKE P+564,16:POKE P+565,16:POKE P+566,16:POKE P+567,0
570 PRINT "A LUCKY SYMBOL FOR A WOMAN.  SHE"
580 PRINT "WILL RECEIVE FROM THE OPPOSITE SEX."
590 PRINT "FOR A MAN, JOY IN GIVING."
600 REM GO WAIT FOR KEYSTROKE
610 GOTO 2420
620 PRINT #6;"DARK SIDE OF ARIES"
630 POKE P+560,0:POKE P+561,16:POKE P+562,16:POKE P+563,16
640 POKE P+564,170:POKE P+565,170:POKE P+566,238:POKE P+567,0
650 PRINT "LOSS OR MISPLACEMENT OF"
660 PRINT "A VALUED OBJECT. "
670 PRINT "FRIENDSHIP IN JEOPARDY."
680 GOTO 2420
690 POKE 85,6
700 PRINT #6;TAURUS"
710 POKE P+560,66:POKE P+561,66:POKE P+562,66:POKE P+563,60
720 POKE P+564,36:POKE P+565,66:POKE P+566,66:POKE P+567,60
730 PRINT "LOVE FULFILLED. NOURISHMENT."
740 PRINT " MONEY.  WEALTH. GOOD THINGS."
750 GOTO 2420
760 PRINT #6;"DARK SIDE OF TAURUS"
770 POKE P+560,60:POKE P+561,66:POKE P+562,66:POKE P+563,66
780 POKE P+564,60:POKE P+565,66:POKE P+566,66:POKE P+567,66
790 PRINT "FRUSTRATION.  FINANCIAL"
800 PRINT "SETBACKS.  UNREQUITED LOVE."
810 GOTO 2420
820 POKE 85,6
830  PRINT #6;"GEMINI"
840 POKE P+560,255:POKE P+561,36:POKE P+562,36:POKE P+563,36
850 POKE P+564,36:POKE P+565,36:POKE P+566,36:POKE P+567,255
860 PRINT "A PLEASUREABLE JOURNEY."
870 PRINT "TWO-WAY COMMUNICATION."
880 GOTO 2420
890 PRINT #6;"DARK SIDE OF GEMINI"
900 POKE P+560,255:POKE P+561,36:POKE P+562,36:POKE P+563,36
910 POKE P+564,36:POKE P+565,36:POKE P+566,36:POKE P+567,255
920 PRINT "TRAVEL WILL INTERFERE WITH"
930 PRINT "PLANS TO DEEPEN LOVE."
940 GOTO 2420
950 POKE 85,6
960 PRINT #6;"CANCER"
970 POKE P+560,16:POKE P+561,24:POKE P+562,20:POKE P+563,18:POKE P+564,
144
980 POKE P+565,80:POKE P+566,48:POKE P+567,16
990 PRINT "FAMILY MATTERS, RELATIVES,CHILD."
1000 PRINT "CHILDREN. BIRTH."
1010 GOTO 2420
1020 PRINT #6;"DARK SIDE OF CANCER"
1030 POKE P+560,16:POKE P+561,48:POKE P+562,80:POKE P+563,144
1040 POKE P+564,18:POKE P+565,20:POKE P+566,24:POKE P+567,16
1050 PRINT "STRESSFUL NEWS ABOUT FAMILY."
1060 GOTO 2420
1070 POKE 85,8
1080 PRINT #6;"LEO"
1090 POKE P+560,62:POKE P+561,36:POKE P+562,36:POKE P+563,36
1100 POKE P+564,37:POKE P+565,37:POKE P+566,165:POKE P+567,231
1110 PRINT "COMPLETE TRANSITION. CHANGE OF"
1120 PRINT "ATTITUDE. PROSPERITY. IF BLANK"
1130 PRINT "RUNE NEXT TURN, DEATH . . ."
1140 GOTO 2420
1150 POKE 85,4
1160 PRINT #6;"BLANK RUNE"
```

```
1170 PRINT "   DEATH . . ."
1180 GOTO 2420
1190 POKE 85,7
1200 PRINT #6;"VIRGO"
1210 POKE P+560,8:POKE P+561,4:POKE P+562,18:POKE P+563,33
1220 POKE P+564,64:POKE P+565,132:POKE P+566,72:POKE P+567,32
1230 PRINT "ONE YEAR'S TIME  A HARVEST  DUES PAID"
1240 PRINT "EXPECT ATTORNEYS OR BANKERS."
1250 GOTO 2420
1260 PRINT #6;"THE BLOODY RUNE"
1270 PRINT "BE CAUTIOUS AROUND ENEMIES."
1280 GOTO 2420
1290 POKE 85,7
1300 PRINT #6;"LIBRA"
1310 POKE P+560,126:POKE P+561,66:POKE P+562,36:POKE P+563,231
1320 POKE P+564,0:POKE P+565,255
1330 PRINT "A UNION, UNITING OF SPIRITS FOR A"
1340 PRINT "COMMON GOAL.  GIFTS."
1350 PRINT "GOOD BALANCE. CENTERING"
1360 GOTO 2420
1370 POKE 85,4
1380 PRINT #6;"BLANK RUNE"
1390 PRINT "MYSTERIES HIDDEN BY THE GODS."
1400 PRINT "SECRETS BETTER LEFT UNTOLD."
1410 GOTO 2420
1420 POKE 85,5
1430 PRINT #6;"SCORPIO"
1440 POKE P+560,193:POKE P+561,162:POKE P+562,156:POKE P+563,128
1450 POKE P+564,128:POKE P+565,156:POKE P+566,162:POKE P+567,193
1460 PRINT "UNEXPECTED MATERIAL WEALTH.  GAINS."
1470 PRINT "SECRETS.  MYSTIQUE."
1480 GOTO 2420
1490  PRINT #6;"DARK SIDE OF SCORPIO"
1500 POKE P+560,193:POKE P+561,162:POKE P+562,156:POKE P+563,128
1510 POKE P+564,128:POKE P+565,156:POKE P+566,162:POKE P+567,193
1520 PRINT "YOU HAVE EXPECTED TOO MUCH AND WILL"
1530 PRINT "BE DISAPPOINTED. "
1540 PRINT "COURAGE."
1550 GOTO 2420
1560 POKE 85,5
1570 PRINT #6;"SAGITTARIUS"
1580 POKE P+560,16:POKE P+561,56:POKE P+562,84:POKE P+563,146
1590 POKE P+564,16:POKE P+565,16:POKE P+566,124:POKE P+567,16
1600 PRINT "INCONVENIENCE.  ALL WILL WORK OUT "
1610 PRINT "ACCORDING TO PLAN.  MAYBE NOT YOUR"
1620 PRINT "PLAN, BUT ACCORDING TO PLAN."
1630 GOTO 2420
1640 POKE 85,3
1650 PRINT #6;"CAPRICORN"
1660 POKE P+560,255:POKE P+561,68:POKE P+562,40:POKE P+563,16
1670 POKE P+564,68:POKE P+565,255
1680 PRINT "INHERITANCE.  A WILL.  OLDER FOLKS"
1690 PRINT "OF KINDLY DISPOSITION."
1700 GOTO 2420
1710 PRINT #6;"DARK SIDE OF LIBRA"
1720 POKE P+560,255:POKE P+561,0:POKE P+562,231:POKE P+563,36
1730 POKE P+564,66:POKE P+565,126
1740 PRINT "      A GIFT."
1750 GOTO 2420
1760 POKE 85,4
1770 PRINT #6;"BLANK RUNE"
1780 GOTO 2420
1790 PRINT #6;"DARK SIDE CAPRICORN"
1800 POKE P+560,16:POKE P+561,40:POKE P+562,68:POKE P+563,130
```

```
1810 POKE P+564,68:POKE P+565,40:POKE P+566,16:POKE P+567,40
1820 PRINT "WATCH OUT FOR MACHINES."
1830 PRINT "BE AWARE! MECHANICAL DEVICES CAN"
1840 PRINT "CAUSE ACCIDENTS."
1850 GOTO 2420
1860 POKE 85,4
1870 PRINT #6;"AQUARIUS"
1880 POKE P+560,33:POKE P+561,170:POKE P+562,68:POKE P+563,0
1890 POKE P+564,17:POKE P+565,42:POKE P+566,68
1900 PRINT "BODILY WELFARE IMPORTANT. NEW"
1910 PRINT "AGE AWARENESS. FULFILL HUMANISTIC"
1920 PRINT "MATTERS. WATCH BIORHYTHMS."
1930 GOTO 2420
1940 PRINT  #6;"DARK SIDE AQUARIUS"
1950 POKE P+560,33:POKE P+561,170:POKE P+562,68:POKE P+563,0
1960 POKE P+564,17:POKE P+565,42:POKE P+566,68
1970 PRINT "YOU HAVE AN ENEMY.  THE NEXT RUNE"
1980 PRINT "WILL TELL HOW TO DEAL WITH THEM"
1990 GOTO 2420
2000 POKE 85,6
2010 PRINT #6;"PISCES"
2020 POKE P+560,54:POKE P+561,20:POKE P+562,20:POKE P+563,123
2030 POKE P+564,20:POKE P+565,20:POKE P+566,54
2040 PRINT "NEW AND STIMULATING ENVIRONMENT!"
2050 PRINT "YOUR MIND WILL FLOURISH"
2060 GOTO 2420
2070 PRINT #6;"DARK SIDE OF PISCES"
2080 POKE P+560,54:POKE P+561,20:POKE P+562,20
2090 POKE P+563,123:POKE P+564,20:POKE P+564,20:POKE P+565,54
2100 PRINT "DON'T GET INVOLVED WITH"
2110 PRINT "ONES WHO WILL USE YOU."
2120 GOTO 2420
2130 PRINT #6;"DARK SIDE OF LEO"
2140 POKE P+560,231:POKE P+561,165:POKE P+562,37:POKE P+563,37
2150 POKE P+564,36:POKE P+565,36:POKE P+566,36:POKE P+567,62
2160 PRINT "PROSPERITY. CLEAN AIR.  FRESH WATER."
2170 GOTO 2420
2180 PRINT #6;"DARK SIDE OF VIRGO"
2190 POKE P+560,254:POKE P+561,68:POKE P+562,40:POKE P+563,16
2200 POKE P+564,40:POKE P+565,68:POKE P+566,254
2210 PRINT "ONE YEAR.  A HARVEST."
2220 GOTO 2420
2230 PRINT #6;"   DARK   SIDE  OF"
2240 PRINT #6;""
2250 PRINT #6;"   SAGITTARIUS"
2260 POKE P+560,16:POKE P+561,124:POKE P+562,16:POKE P+563,16
2270 POKE P+564,146:POKE P+565,84:POKE P+566,56:POKE P+567,16
2280 PRINT "HUNTING GOD, AROOLEW."
2290 PRINT "AN ELK. POWER."
2300 GOTO 2420
2310 POKE 85,8
2230 PRINT #6;"ODIN"
2330 POKE P+560,137:POKE P+561,74:POKE P+562,60:POKE P+563,8
2340 POKE P+564,8:POKE P+565,8:POKE P+566,8:POKE P+567,8
2350 PRINT "BIRCH  TWIG. FERTILITY. GROWTH."
2360 PRINT "HEALTH.  PROTECTION."
2370 PRINT "A MAN."
2380 GOTO 2420
2390 POKE 85,7
2400 PRINT #6;"TORCH"
2410 PRINT "FIRE. RAGE. PASSION."
2420 REM HIDE CURSOR
2430 POKE 755,0
2440 REM WAIT FOR KEYSTROKE
```

```
2450 GOSUB 2800
2460 REM PUT OLD SYMBOL OFF SCREEN
2470 POKE 53248,0
2480 POKE 53277,0:REM TURN OFF GRAPHICS
2490 GRAPHICS 2+16:REM CLEAR SCREEN
2500 GOTO 310:REM GET NEW SYMBOL
2510 GRAPHICS 0:REM CLEAR SCREEN
2520 POKE 84,3:REM SET VERT. MARGIN
2530 REM PRINT INSTRUCTIONS.
2540 PRINT "I AM THE DELPHIC ORACLE."
2550 PRINT
2560 PRINT "TOGETHER WE WILL FORTELL THE"
2570 PRINT "EVENTS OF TOMMORROW."
2580 PRINT
2590 PRINT "TO PLAY THE ORACLE . . .   HIT"
2600 PRINT "ANY  OF THE KEYS.  WHEN YOU DO,"
2610 PRINT "A SYMBOL WILL APPEAR ON THE SCREEN"
2620 PRINT "ALONG WITH A MESSAGE."
2630 PRINT
2640 PRINT "UP TO THREE SEPARATE SYMBOLS MAY BE"
2650 PRINT "REQUIRED TO GIVE ENOUGH INFORMATION"
2660 PRINT "TO TELL YOU WHAT WILL BE.":PRINT
2670 PRINT
2680 PRINT "IT WORKS BEST TO CONCENTRATE ON"
2690 PRINT "A QUESTION BEFORE KEYING THE SET OF"
2700 PRINT "THREE SYMBOLS."
2710 POSITION 5,23
2720 PRINT "(HIT ANY KEY TO CONTINUE)"
2730  POKE 755,0:REM HIDE CURSOR
2740 REM WAIT FOR KEY TO BE PRESSED
2750 GOSUB 2800
2760 PRINT CHR$(125):REM CLEAR SCREEN
2770 REM GO PLAY GAME
2780 GOTO 310
2790 REM WAIT FOR KEYSTROKE
2800 POKE 764,255
2810 IF PEEK(764)=255 THEN 2810
2820 POKE 764,255
2830 RETURN
```

15 / I-Ching Coin Toss

The I-Ching, pronounced *yee-ching*, is an ancient Chinese method for determining one's fate. Pre-Computer Age practitioners used a set of tortoise-shell coins to determine a pattern of solid and broken lines. These patterns were analyzed and the verdict decreed.

We'll use the ATARI Home Computer to cast the patterns. You'll also need a compendium of I-Ching interpretations. There are a number of versions available in bookstores.

This computer version of the I-Ching prints two sets of color bars on the screen. They represent the result of tossing those ancient coins. Take the results and consult an I-Ching text to determine the answer to your question.

To toss the coins, RUN the program. It will display two sets of three broken and solid lines. These colored lines represent the tossing of the coins.

To toss again, hit any key. Every time a key is touched, the program will toss a new set of coins.

How the Program Works

Line 20 blanks the screen. Line 30 chooses Graphics Mode 2, full screen.

Line 40 paints the screen.

Lines 60 through 80 format and print the program title.

Line 110 goes to the Player/Missile graphics subroutine.

Lines 140 through 320 choose the lines that will represent the coin toss. Lines 170, 180, 210, 240, 270, and 300 each choose a random number, either a 1 or a 2. In all cases if the random number is a 1, then the variables A through F will equal 231. If it's a 2, by default, those variables are tagged with the value 255.

Lines 340 through 390 draw six lines. Each line is either solid or broken. That's determined by the value saved in the random numbers generated in variables A through F. Those values are poked into the memory where Player/Missile graphics are drawn.

Line 410 displays those coins in the center of the screen.

Lines 430 through 450 wait for any key to be pressed, in order to draw the next set of coins tossed.

Player/Missile Graphics

Lines 480 through 610 do the drawing.

Line 490 gets the top RAM address. Line 500 tells the graphics chip where to draw. The poke 559,46 tells the system to draw in double-line resolution.

Lines 520 and 530 fill the graphics memory with zeros, to erase anything possibly left over from a previous program.

Line 550 draws the coins to the left, just offscreen. Line 570 draws them extra wide.

Line 590 paints the coins yellow. Line 610 turns on the graphics.

String and Numeric Variables

TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.
MEM MEM	used in loop to plant zeros in memory. A through F random numbers, either a 1 or a 2.
FIRST	through SIXTH horizontal bars equal either 231 or 255, broken or solid lines.

The Program

```
10 REM I- CHING COIN TOSS
20 PRINT CHR$(125):REM CLEAR SCREEN
30 GRAPHICS 2+16:REM CHOOSE GRAPHICS
40 SETCOLOR 4,8,4:REM COLOR SCREEN
50 REM LOCATE CURSOR
60 POSITION 5,0
70 REM PRINT BIG LETTERS
80 PRINT #6;"I - CHING"
90 REM GO DRAW COINS
100 REM GO INITIALIZE GRAPHICS
110 GOSUB 490
120 REM GET RANDOM NUMBER
130 REM ONE = BROKEN BAR, OTHERWISE SOLID
140 LET A=INT(2*RND(1))
150 REM BAR IS SOLID UNLESS A = 1
160 FIRST=255
170 IF A=1 THEN FIRST=231
180 LET B=INT(2*RND(1))+1
190 SECOND=255
200 IF B=1 THEN SECOND=231
210 LET C=INT(2*RND(1))+1
220 THIRD=255
230 IF C=1 THEN THIRD=231
240 LET D=INT(2*RND(1))+1
250 FOURTH=255
260 IF D=1 THEN FOURTH=231
270 LET E=INT(2*RND(1))+1
280 FIFTH=255
290 IF E=1 THEN FIFTH=231
300 LET F=INT(2*RND(1))+1
310 SIXTH=255
320 IF F=1 THEN SIXTH=231
330 REM FILL IN EACH OF SIX BARS
340 POKE PMBASE+573,FIRST
350 POKE PMBASE+576,SECOND
360 POKE PMBASE+579,THIRD
370 POKE PMBASE+700,FOURTH
380 POKE PMBASE+703,FIFTH
390 POKE PMBASE+706,SIXTH
400 REM PLACE COINS ON SCREEN
410 POKE 53248,90:POKE 53249,130
420 REM WAIT FOR KEY TO BE TOUCHED
430 POKE 764,255
440 IF PEEK(764)=255 THEN 440
450 POKE 764,255
460 REM DO NEXT COIN TOSS
470 GOTO 140
480 REM INITIALIZE PLAYER/MISSILE GRAPHICS
490 TOP=PEEK(106)-8:POKE 54279,TOP
500 PMBASE=256*TOP:POKE 559,46
510 REM CLEAR MEMORY WITH ZEROS
520 FOR MEM=PMBASE+384 TO PMBASE+1024
530 POKE MEM,0:NEXT MEM
540 REM PLACE BARS OFF-SCREEN
550 POKE 53248,10:POKE 53249,10
560 REM SET PLAYER SIZE
570 POKE 53256,3:POKE 53257,3
580 REM PAINT COINS
590 POKE 704,255:POKE 705,255
600 REM TURN ON GRAPHICS SWITCH
610 POKE 53277,3
620 RETURN
```

16 / Adventure Dice Cast

Dungeon Masters can ease the task of assigning player strength, intelligence, karma, weapons, and other characteristics. This is a casting program for 4-, 6-, 8-, 10-, 12-, and 20-sided dice. Select a number from the menu to tell the program which dice to cast, and it instantly tells you the result. To cast a new number, choose the appropriate number representing the dice and a new number is cast.

How the Program Works

Lines 130 through 330 format and print the menu.

Line 340 opens the keyboard for a read. This way, when any key is hit, it triggers a response. You won't have to be bothered with hitting return. Do be sure to enter only the number 1 through 6, though.

Line 360 gets your choice. The ATARI Home Computer reads it in its ATASCII code. The number 1, for instance, is 49 in ATASCII.

Line 390 converts the ATASCII roll to a branching routine.

Lines 410 through 520 roll the dice.

Lines 540 through 580 print the results and reload the dice.

String and Numeric Variables

ROLL	The roll of the dice.
BLANK$	One blank space, used in printing the result of the roll.

The Program

```
100 REM ADVENTURE DICE CAST
110 DIM BLANK$(1):BLANK$=" ":REM ONE SPACE
120 PRINT CHR$(125):REM CLEAR SCREEN
130 POKE 84,1:REM SET VERT. MARGIN
140 REM PRINT MESSAGE
150 PRINT "HOW MANY SIDED DICE TO CAST?"
160 REM LOCATE CURSOR
170 POSITION 5,3
180 PRINT "1. FOUR"
190 REM LOCATE CURSOR
200 POSITION 5,5
210 PRINT "2. SIX"
220 POSITION 5,7
230 PRINT "3. EIGHT"
240 POSITION 5,9
250 PRINT "4. TEN"
260 POSITION 5,11
280 PRINT "5. TWELVE"
290 POSITION 5,13
300 PRINT "6. TWENTY"
310 POKE 84,15
320 PRINT "SELECT A NUMBER (1 - 6)"
330 POKE 755,0:REM HIDE CURSOR
340 OPEN #1,4,0,"K:"
350 REM WHICH NUMBER PRESSED
360 GET #1,ROLL
370 REM TURN OFF KEYBOARD
380 REM GO THROW DICE
390 ON ROLL-48 GOTO 420,440,460,480,500,520
400 GOTO 100
410 REM CAST DIE
420 ROLL=INT(4*RND(1))+1
430 GOTO 530
440 ROLL=INT(6*RND(1))+1
450 GOTO 530
460 ROLL=INT(8*RND(1))+1
470 GOTO 530
480 ROLL=INT(10*RND(1))+1
490 GOTO 530
500 ROLL=INT(12*RND(1))+1
510 GOTO 530
520 ROLL=INT(20*RND(1))+1
530 REM LOCATE CURSOR
540 POSITION 10,20
550 REM PRINT RESULTS
560 PRINT "DIE CAST  ";ROLL;BLANK$
570 REM DO NEXT TOSS
580 GOTO 360
```

17 / "R" Is for Red

Have you ever watched kids and grownups at a video arcade pouring a fortune's worth of quarters into games of intergalactic conquest? Besides passing time and building hand-eye coordination, the gamers are acquiring an advantage unavailable to yesteryear's Neanderthal. With our readily accessible micros comes a new way of thinking, seeing, and communicating with the universe. And, although at first glance the "R" Is for Red game may appear similar to a well-known chainstore home entertainment device, close scrutiny may reveal several dimensions that run deeper than any party game.

One of the discoveries in this dawn of high technology is that different hemispheres of the brain click in or out while different tasks are performed. In this light and sound cryptogram, there are two different dimensions in which the brain can operate. First, is the conscious level, where one can remember—"Let's see, I saw a blue bar, two greens, and then a red." But probably more important is the unconscious, or Zen, part—the Force part where one does not think about the tennis, jogging, or computer game strategy. One simply reacts without thinking, like so many of us do while performing a very familiar task. In "R" Is for Red, the color and music sequence is not remembered for its individual sight and sound, but rather is recalled as an entire block, or word if you will, made up of colors. The color bars are as an alphabet, where words are not spelled out with A to Z characters, but instead with patterns of blues, reds, greens, and yellows. As the colors flash on the screen, one does not dissect the word letter-by-letter, but speed reads it as a word, phrase, and sentence.

Sound fascinating? Recall the film *Close Encounters of the* Third Kind, where an international team of scientists assembled at

Devil's Tower to set up a music synthesizer/light show to communicate with aliens. They communicated with blocks of color and sound in a high tech, computer-driven Esperanto. How many colors can your eyes and brain distinguish? Six or seven at one time is about average, but with practice, the galaxy is the limit.

One of the more sophisticated software aspects of this color cryptogram is how it uses the Player/Missile graphics. While no animation is used, the program does make use of some of the unique qualities of the ATARI Home Computer. In this case, speed. Instead of having to draw, undraw, then draw again every time the program needs to flash a color on the screen, the ATARI Home Computer uses the Player/Missile graphics. The first step is to draw the bars from top to bottom, one at a time, from left to right. Since no color for the bars is specified at the time of drawing, they are automatically assigned the default color, black.

All four players are displayed on the screen, side by side, waiting for the routine that will call them to life when the appropriate color register is poked. It remains painted for as long as the skill level delay loop lets it. The bar then fades to black, and the next appropriate color bar comes out of hiding.

With the ATARI Home Computer, each player has an individual-size register that controls width. For the sake of experimentation, poke the number 3 instead of the number 1 with each of the four registers assigned to statement number 1280. RUN the program. Each player will have quadrupled in width, and each will overlap another. In fact, the program will now look quite ugly. For an aesthetic remedy, change the horizontal locations of each of the players in lines 1300 and 1310. Replace the values of 90,110,130 and 150 with 50,90,130 and 170, respectively. The next time the program is run, it will look much prettier.

To play the game, wait while the Player/Missile graphics initialize. Read, filling the graphics memory RAM with zeros to clear it before the program actually draws the color bars. See how this is done in lines 1160 through 1260. Once the micro has finished drawing the color bars, a message will flash on the screen: SKILL LEVEL (1-5). At that point, hit any number from 1 to 5, with 1 being the slowest and easiest routine, and 5 being the fastest and toughest. By the way, 5 is also the most interesting, in terms of video and audio.

As the program puts you through your paces, it will flash a color bar on the screen, the color accompanied by its own musical note. When you see a blue bar, hit the letter B. Just touch the key; you won't need to hit RETURN. R is for red, B is for blue, and Y is for yellow. When you respond with the correct key for the correct color, another color/musical note will be added to the sequence. Get the picture?

How the Program Works

Line 110 clears the screen.

Line 120 directs the program to the Player/Missile graphics sequence.

Lines 1190 through 1260 draw the vertical bars 70-bits tall by 8-bits wide and paint them black, by default. Line 1280 sets the player width at single wide, which translates into 8-bits across.

Lines 1300 through 1310 place the boxes at their horizontal positions.

Line 1330 gives the final signal to turn on the graphics.

Meanwhile back at the top of the program: Line 170 prints the skill-level message, and the micro waits at line 200 for the choice. When a key from number 1 through 5 is typed, that key's ATASCII code is compared to the table in lines 230 to 270. The appropriate skill value will be placed in the delay loop that determines how long the color bars will stretch on screen. Line 290 routes us to the beginning of the main program loop.

Line 640 picks a random number, any number between 1 and 4. That number is assigned a number. The first number will be between 1 and 4 and will tell the micro which of the four colors—red, blue, green, or yellow—to display. The second number, which keeps track of each particular color in the sequence, is assigned in each turn by line 690. With a color chosen, and having told the micro where it is in the sequence, the program jumps to line 300.

Lines 320 to 410 hold the program loop where all the colors in the sequence so far are flashed on the screen. You will see all the previous colors you have remembered and keyed correctly, plus one addition to the pattern.

Lines 470 through 600 will test your memory of the sequence.

As each color comes up in its turn, 340 through 370 will send the program to the particular subroutine that paints that color. There are four such subroutines: lines 720 to 770, 780 to 830, 840 to 890, and finally, 900 to 950. Once at one of these subroutines, the program will paint the bar either red, blue, green, or yellow. Then comes the musical note. Color and note are delayed for as long as was specified in the skill level. After the delay is satisfied, the sound is killed and the bar is painted black; then it's back to the main loop for the next color bar.

The loop knows how many colors were flashed on the screen and will wait for that many keystrokes. Each keystroke will be matched against the proper sequence. If you enter a color out of sequence, the program lumbers off to the blunder routine to let you know you aren't as quick or as smart as you once were.

The blunder routine, Lines 1000 through 1070, shuts off the

keyboards, plays a sour note, clears the value of all variables to zero, and starts the game over.

However, if you haven't yet goofed in remembering the color sequence, the game will continue adding random color bars to the sequence. One thought: Why not close both eyes and play by ear? Can you recall the sequence of musical notes? How about manual dexterity? Where is that "R"? Here is where one tries out the computer Zen mentioned earlier, remembering what we "saw" with the mind's eye, reaching out with that inner sanctum of grey matter that we little understand and so much crave to master. After all, that's why we're into micros, right?

String and Numeric Variables

SKILL	Skill level chosen. Holds ATASCII number for 1 through 5. Then assigns one of five numbers, 10, 25, 50, 100, 150.
COUNT	The color of the next color bar to be flashed on the screen.
NUMBER	Number of color bars flashed on screen.
RESPONSE	Which color bar. "R" is for red. Holds ATASCII number for each letter.
ANSWER	Which color bar.
COLOR	Random number used to choose color of bars.
I	Holds value of sound delay loops.
TURN	Where in sequence of color bars.
TOP	Memory address 2K below top of RAM.
PMBASE	Pointer to beginning of Player/Missile memory.
RED, BLUE, GREEN, YELLOW	Where each respective box (Player/Missile) is drawn in memory.

The Program

```
100 REM "R" IS FOR RED GAME
110 ? CHR$(125):REM CLEAR SCREEN
120 GOSUB 1080:REM GO DRAW COLOR BARS
130 DIM TURN(255),SKILL(3),NUMBER(255)
140 REM LOCATE CURSOR
150 POSITION 2,10
```

```
160 REM CHOOSE SKILL LEVEL
170 ? #6;"SKILL LEVEL  (1 - 5)"
180 REM READ KEYBOARD
190 OPEN #1,12,0,"K"
200 GET #1,SKILL
210 CLOSE #1
220 REM SET SPEED OF DISPLAY
230 IF SKILL=49 THEN SKILL=150
240 IF SKILL=50 THEN SKILL=100
250 IF SKILL=51 THEN SKILL=50
260 IF SKILL=52 THEN SKILL=25
270 IF SKILL=53 THEN SKILL=10
280 REM GET FIRST COLOUR
290 GOTO 640
300 COUNT=0:REM RESET COUNTER
310 REM LOOP TO DO EACH COLOUR
320 FOR I=1 TO NUMBER
330 REM DISPLAY EACH COLOR
340 IF TURN(COUNT)=0 THEN GOSUB 720
350 IF TURN(COUNT)=1 THEN GOSUB 780
360 IF TURN(COUNT)=2 THEN GOSUB 840
370 IF TURN(COUNT)=3 THEN GOSUB 900
380 REM INCREMENT EACH TIME
390 COUNT=COUNT+1
400 REM DO NEXT COLOR
410 NEXT I
420 REM OPEN KEYBOARD  FOR READ
430 OPEN #1,4,0,"K:"
440 COUNT=0
450 REM FIRST TIME THROUGH IS 1
460 IF NUMBER<1 THEN NUMBER=1
470 FOR K=1 TO NUMBER
480 REM READ KEYBOARD
490 GET #1,RESPONSE
500 REM KEEP TRACK OF TIMES THROUGH
510 LET ANSWER=TURN(COUNT)
520 REM TRANSLATE ANSWER
530 IF ANSWER=0 THEN ANSWER=82
540 IF ANSWER=1 THEN ANSWER=66
550 IF ANSWER=2 THEN ANSWER=71
560 IF ANSWER=3 THEN ANSWER=89
570 REM MATCH PATTERN AGAINST PLAYER RESPONSE
580 IF RESPONSE<>ANSWER THEN 970
590 COUNT=COUNT+1
600 NEXT K
610 CLOSE #1
620 NUMBER=NUMBER+1
630 REM CHOOSE RANDOM COLOUR
640 LET COLOR=INT(4*RND(1))
650 REM SAVE COUNT IN NUMBER
660 LET COUNT=NUMBER
670 IF COUNT=0 THEN COUNT=1
680 REM LABLE EACH COLOR IN SEQUENCE
690 LET TURN(COUNT)=COLOR
700 REM DO AGAIN
710 GOTO 300
720 POKE 704,48:REM COLOR RED
730 REM MAKE SOUND
740 SOUND 1,50,10,8:FOR J=1 TO SKILL:NEXT J
750 SOUND 1,0,0,0:REM KILL SOUND
760 POKE 704,0:REM COLOR BLACK
770 RETURN
780 POKE 705,128:REM COLOR BLUE
790 REM MAKE SOUND
```

```
800 SOUND 1,72,10,8:FOR J=1 TO SKILL:NEXT J
810 SOUND 1,0,0,0:REM KILL SOUND
820 POKE 705,0:REM COLOR BLACK
830 RETURN
840 POKE 706,192:REM COLOR GREEN
850 REM MAKE SOUND
860 SOUND 3,96,10,8:FOR J=1 TO SKILL:NEXT J
870 SOUND 3,0,0,0:REM KILL SOUND
880 POKE 706,0:REM COLOR BLACK
890 RETURN
900 POKE 707,255:REM COLOR YELLOW
910 REM MAKE SOUND
920 SOUND 3,96,10,8:FOR J=1 TO SKILL:NEXT J
930 SOUND 3,0,0,0:REM KILL SOUND
940 POKE 707,0:REM COLOR BLACK
950 RETURN
960 TRAP 960:GOTO 240
970 REM MISTAKE ROUTINE
980 CLOSE #1:REM TURN-OFF KEYBOARD
990 REM BLUNDER SOUND
1000 SOUND 1,243,10,8
1010 REM DELAY TO PLAY SOUND
1020 FOR I=1 TO 200:NEXT I
1030 REM TURN OFF SOUND
1040 SOUND 1,0,0,0
1050 REM CLEAR ALL VARIABLES TO ZERO
1060 REM THEN BEGIN AGAIN
1070 CLR :GOTO 130
1080 GRAPHICS 2+16:SETCOLOR 4,7,4
1090 REM LOCATE CURSOR
1100 POSITION 4,10
1110 REM WAIT MESSAGE WHILE DRAWING
1120 REM PLAYER/MISSILE GRAPHICS
1130 ? #6;"PLEASE WAIT"
1140 TOP=PEEK(106)-8:POKE 54279,TOP
1150 PMBASE=256*TOP:POKE 559,46
1160 REM CLEAR GRAPHIC MEMORY BY LOADING WITH ZEROS
1170 FOR I=PMBASE+512 TO PMBASE+1024:POKE I,0:NEXT I
1180 REM DRAW BOXES 8 BITS WIDE BY 70 BITS TALL
1190 FOR RED=PMBASE+528 TO PMBASE+598
1200 POKE RED,255:NEXT RED
1210 FOR BLUE=PMBASE+656 TO PMBASE+726
1220 POKE BLUE,255:NEXT BLUE
1230 FOR GREEN=PMBASE+784 TO PMBASE+854
1240 POKE GREEN,255:NEXT GREEN
1250 FOR YELLOW=PMBASE+912 TO PMBASE+982
1260 POKE YELLOW,255:NEXT YELLOW
1270 REM SET WIDTH OF BOXES
1280 POKE 53256,1:POKE 53257,1:POKE 53258,1:POKE 53259,1
1290 REM LOCATE BOXES HORIZ. POSITION
1300 POKE 53248,90:POKE 53249,110
1310 POKE 53250,130:POKE 53251,150
1320 REM TURN ON GRAPHICS
1330 POKE 53277,3
1340 RETURN
```

18 / Car Ownership

Are you considering buying another car? How much will it really cost to own, per year, per month, and per mile? The costs will be of two types: fixed and variable. Fixed costs include yearly depreciation of the vehicle, the installment loan, interest, insurance, taxes, license plates, and municipal stickers. Variable costs include fuel, oil and grease, tires, batteries, windshield wipers, burnt-out headlights, tolls, and the Saturday afternoon wash'n'wax job.

Fixed costs are hard to allocate. For instance, if fixed charges total $1,000 a year, and the car is driven 24,000 miles that year, fixed costs total four cents a mile. But if you drive only 5,000 miles in a year, fixed costs jump to twenty cents a mile.

One thing is certain: the cost of owning and operating a well-maintained car goes down sharply as it gets older, especially after the first year. This is primarily due to a decrease in depreciation. The drop in fixed charges, however, is partially offset by an increase in maintenance costs. Nevertheless, for a compact car, the cost per mile falls from about 58 cents in its first year—including fixed and variable costs—to about 22 cents on its tenth birthday.

Much of the car's total cost over ten years, depends on trade-in allowance for the old heap, actual price paid for the car, interest rate, and the depreciation schedule set up with the car's anticipated use.

Fuel cost is another important consideration in either buying a new car or considering selling an old one. If a particular model gets 24 MPG, and another gets 14 MPG, assuming fuel costs $1.25 a gallon, and each car is driven 10,000 miles, fuel will cost $296 less for the more fuel-efficient vehicle. Over a long period of time, that amounts to a sizeable sum.

Standard transmissions normally get better gas mileage because the engine is not straining at a stop sign to move the car forward. A standard transmission idles the engine with the transmission out of gear. However, poor driving habits can blow that advantage.

Axle ratios affect how many engine revolutions are needed to turn the wheel one full turn. Low axle ratios save gas and keep the engine running at a lower RPM. High ratios gear the engine down for pulling trailers or hauling "toppers" (top part of a camper).

Diesel engines can reward drivers with an increase in mileage of as much as 25 percent, but they are more expensive to purchase and maintain. One has to drive lots of miles to make the additional up-front expense pay off. Besides the monetary expense, one should consider the controversy over whether or not diesel engines emit cancer-causing pollutants.

Tires affect mileage. Low pressure increases road resistance, and thus gasoline consumption. Keep tires properly inflated. It's a good idea to invest in your own tire gauge, so you won't have to face those grumpy service station personnel who only begrudgingly hand over a filthy gauge and stare at you. Radial tires cost more initially, but deliver 3–8 percent better mileage. Plus they last longer and improve handling.

Cruise control can eliminate a driver's inconsistency, by keeping the car moving at a relatively constant velocity—without those inadvertent leg shifts.

Air conditioners use lots of engine horsepower, typically pulling down gas mileage 1–3 MPG. Is it worth it? Power seats, antennas, brakes, and steering also rob fuel dollars.

The biggest waste of fuel is an out-of-tune engine. A tune-up usually pays for itself in a few tankfuls.

How the Program Works

Lines 150 through 430 ask for the fixed-cost particulars—how many MPG, cost of loan payments, and the like. The answers are stored in descriptive numeric variables.

Lines 460 and 480 perform preliminary calculations.

Lines 500 and 510 tally all of the fixed costs, storing the sum in the variable TOTAL.

Lines 530 through 610 break down the cost per mile, per month, and per year.

Lines 630 through 770 format and print the results.

String and Numeric Variables

MILEAGE	Average miles driven per year.
MPG	Miles per gallon.
PRICE	Cost per gallon of fuel.
LOAN	Amount of monthly car payment (enter 0 if none).
INSURANCE	Cost to insure vehicle for one year.
TUNE	Money usually spent in a year on engine work.
SHOCKS	A new set needed?
TIRES	New tires needed? Snow tires?
LUBE	Amount usually spent over a year on oil changes and grease jobs.
MISC	Amount spent on windshield wiper blades, burnt-out tail lights, and so on.
TOTAL	Total sum of expenses.
PERMILE	Cost per mile to operate car.
PERMONTH	Average costs per month.
PERYEAR	Average cost per year.

The Program

```
100 REM CAR COST TALLY
110 PRINT CHR$(125):REM CLEAR SCREEN
120 POKE 85,9
130 PRINT "CAR COST CALCULATOR"
140 POKE 84,3
150 PRINT "MILES DRIVEN PER YEAR"
160 INPUT MILEAGE
170 PRINT
180 PRINT "MILES TO THE GALLON"
190 INPUT MPG
200 PRINT
210 PRINT "AVERAGE PRICE PER GALLON"
220 INPUT PRICE
230 PRINT
240 PRINT "MONTHLY LOAN PAYMENT"
250 INPUT LOAN
260 PRINT
270 PRINT "INSURANCE COST (1 YR.)"
280 INPUT INSURANCE
290 PRINT
300 PRINT "COST OF TUNE-UP"
310 INPUT TUNE
```

```
320 PRINT
330 PRINT "SHOCK ABSORBERS"
340 INPUT SHOCKS
350 PRINT
360 PRINT "NEW TIRES"
370 INPUT TIRES
380 PRINT
390 PRINT "OIL CHANGES & LUBE"
400 INPUT LUBE
410 PRINT
420 PRINT "MISC. EXPENSES"
430 INPUT MISC
440 PRINT
450 REM CALCULATE COSTS
460 LOAN=LOAN*12
470 REM TOTAL GALLONS FUEL FOR YEAR
480 GALLONS=MILEAGE/MPG
490 REM COST OF FUEL FOR YEAR
500 FUELCOST=INT(((PRICE*GALLONS)*100)+0.5)/100
510 TOTAL=FUELCOST+LOAN+INSURANCE+TUNE+SHOCKS+TIRES+LUBE+MISC
520 REM CONVERT TO TWO PLACE DECIMAL
530 TOTAL=INT((TOTAL*100)+0.5)/100
540 REM FIGURE PER MILE COST
550 PERMILE=TOTAL/MILEAGE
560 REM CONVERT TO TWO DECIMAL PLACE
570 PERMILE=INT((PERMILE*100)+0.5)/100
580 REM FIGURE PER MONTH COST
590 PERMONTH=TOTAL/12
600 REM CONVERT TO TWO DECIMAL PLACE
610 PERMONTH=INT((PERMONTH*100)+0.5)/100
620 REM PRINT RESULTS
630 PRINT CHR$(125):REM CLEAR SCREEN
640 POSITION 8,3
650 PRINT "ANNUAL OPERATING COST"
660 PRINT
670 PRINT
680 PRINT "COST PER MILE","$";PERMILE
690 PRINT
700 PRINT "COST PER MONTH","$";PERMONTH
710 PRINT
720 PRINT "FUEL FOR YEAR","$";FUELCOST
730 PRINT
740 PRINT "_____"
750 PRINT
760 PRINT ,"TOTAL","$";TOTAL
770 POSITION 1,20
```

19 / Trip Cost Tabulator

Here's a simple way to plan a trip and know ahead of time how much money you'll need to enjoy the journey. Plug in the miles you'll drive, round trip. If there'll be a lot of sightseeing once you get there, don't forget to tally in those extra miles.

Does the engine burn oil? If so, you can calculate that additional cost.

Next, how many miles per gallon does the engine get? Remember, EPA ratings are only a general indication of how well, or poorly, a car will perform. That kind of information is best gathered over a couple of months with scrupulous record-keeping. Perhaps you'd like to design a simple program for encoding those results.

How much will fuel cost on the road? Truck stops and other fueling stations located on well-traveled highways tend to charge inflated prices. Gasoline may be a nickel or a dime per gallon cheaper in your neighborhood than somewhere on Interstate 80. Plan ahead.

If you are going to sleep in a motel and enjoy the conveniences of a real bed and a sauna, include the average price you'll be paying for a room. If you will camp out along the way, type in that fee.

How many days will you be away from home? How much will be budgeted for food per day?

With all the variables entered, the screen will clear and display a breakdown of the trip. You'll know at a glance how much fuel and oil, lodging, and food will cost. Further breakdowns will include the total cost, and the per mile and per day cost.

How the Program Works

Line 120 blanks the screen.

Typically this program consists of a prompt such as:

```
ROUND TRIP MILES?
```

Inputting the information stores it in a numeric variable. This is true of MPG, motel prices, and so on. Later, in line 410, the calculation begins. With all of the information entered, line 410 will figure out how much it will cost to drive the car from point A to point B.

Line 450 figures out how much it will cost to stay in a motel.

Line 470 figures out how much food will cost for the total number of days you'll be gone.

Line 490 totals all of the costs involved.

Line 530 breaks it down into a PERDAY figure.

Line 570 calculates the PERMILE figure.

Line 580 blanks the screen.

Lines 590 through 750 format and display the cost breakdowns.

String and Numeric Variables

MILES	Round trip miles.
A$	Yes or No. (Engine burns oil?)
MILEAGE	Miles driven per gallon of fuel.
FUELCOST	Cost of a gallon of fuel.
MOTEL	Anticipated cost for one night in motel room.
LODGING	Number of nights to be spent in motel.
DAYS	Total days away from home.
FOOD	Cost of food per day.
TOTAL	Total amount needed for gas, oil, food, and motel.
PERDAY	How much TOTAL averages PERDAY.
PERMILE	Average PERMILE cost.
OIL	Miles driven before quart of oil burned.
OILCOST	Cost of a quart of oil.

The Program

```
100 REM TRIP EXPENSE TABULATOR
110 TRAP 890
120 PRINT CHR$(125):REM CLEAR SCREEN
130 DIM A$(3)
140 POKE 84,15:REM DROP TEXT 15 LINES
150 PRINT "ROUND TRIP MILES"
160 INPUT MILES
170 PRINT
180 PRINT "DOES ENGINE BURN OIL (YES/NO)"
190 INPUT A$
200 REM IF BURNS OIL, GO TALLY COST
210 IF A$="YES" THEN GOSUB 780
220 PRINT
230 PRINT "MILES PER GALLON (HIGHWAY)"
240 INPUT MILEAGE
250 PRINT
260 PRINT "FUEL COST PER GALLON"
270 INPUT FUELCOST
280 PRINT
290 PRINT "AVERAGE PRICE HOTEL/MOTEL ROOM"
300 INPUT MOTEL
310 PRINT
320 PRINT "NIGHTS IN HOTEL/MOTEL ROOM"
330 INPUT LODGING
340 PRINT
350 PRINT "DAYS AWAY FROM HOME"
360 INPUT DAYS
370 PRINT
380 PRINT "FOOD BUDGET PER DAY (DOLLARS)"
390 INPUT FOOD
400 REM CALCULATE TOTAL FUEL COST
410 FUELCOST=(MILES/MILEAGE)*FUELCOST
420 REM ROUND OFF DECIMAL FIGURE
430 FUELCOST = INT((FUELCOST * 100)+.5)/100
440 REM CALCUALTE LODGING EXPENSES
450 LODGING=LODGING*MOTEL
460 REM CALCULATE FOOD EXPENSE
470 FOOD=FOOD*DAYS
480 REM CALCULATE TOTAL TRIP COST
490 TOTAL=FOOD+LODGING+OILCOST+FUELCOST
500 REM ROUND OFF DECIMAL FIGURE
510 TOTAL = INT((TOTAL * 100)+ .5)/100
520 REM TALLY COST PER DAY
530 PERDAY=TOTAL/DAYS
540 REM ROUND OFF DECIMAL FIGURE
550 PERDAY = INT((PERDAY * 100)+ .5)/100
560 REM TALLY COST PER MILE
570 PERMILE=(TOTAL/MILES)
580 PRINT CHR$(125):REM CLEAR SCREEN
590 POKE 84,1:REM SET LEFT MARGIN
600 POKE 85,10:REM DROP TEXT 10 LINES
610 REM PRINT MESSAGE
620 PRINT "TRIP WILL COST:  $"
630 REM LOCATE CURSOR
640 POSITION 28,1
650 REM PRINT TOTAL PRICE OF TRIP
660 PRINT TOTAL
670 POKE 84,5
680 PRINT "FUEL","OIL","LODGING","FOOD"
690 PRINT "$";FUELCOST,"$";OIL,"$";LODGING,"$";FOOD
700 POKE 84,10
```

```
710 PRINT "PER MILE  ( ";MILES;" MILES )"
720 PRINT "$";PERMILE
730 POKE 84,15
740 PRINT "PER DAY  ";"( ";DAYS;" DAYS  )"
750 PRINT "$";PERDAY
760 END
770 REM FIGURE COST OF OIL ON TRIP
780 PRINT
790 PRINT "MILES PER QUART OF OIL"
800 INPUT OIL
810 PRINT
820 PRINT "COST FOR QUART OF OIL"
830 INPUT OILCOST
840 OIL=MILES/OIL
850 OIL=OIL*OILCOST
860 REM BACK TO MAIN PROGRAM
870 RETURN
880 REM IF BAD ENTRY TRY AGAIN
890 TRAP 890:? "PLEASE RE-ENTER . . ."
900 FOR DELAY=1 TO 200:NEXT DELAY:RUN
```

20 / Meal Planner

How much does it cost to cook that exotic quiche your significant lover craves? What about those expensive Austrian pastries you've been aching to try, but were troubled because they might be too expensive to create. Or, just maybe, you'd like to try your hand at the Stroganoff recipe that has been handed down in the family for generations.

No matter what the reason, or the recipe, it's time to get with the program that makes menu-planning easy. Just fill in the blanks. Fill in the number of ingredients you want to use, and, one by one, fill in the cost of each item.

They'll be totaled, and in an instant you'll be apprised of how much an individual serving costs, as well as the total cost of preparing the meal. It's as easy as saying, "Bon appetit!"

How the Program Works

Line 120 blanks the screen.

Line 150 gets the number of ingredients in the recipe. Line 170 will loop back once for each one of those ingredients.

Line 190 asks for the cost of each ingredient. Line 200 assigns that cost to the variable INGREDIENT.

Line 220 adds each INGREDIENT to the sum of COST.

Line 240 keeps the loop going until each ingredient cost is entered.

Line 270 asks for the number of people to be served. Line 280 calculates how much it will cost to feed each person.

Lines 300 through 320 print how much the total meal cost is, as well as the cost per individual serving.

String and Numeric Variables

A	Number of ingredients in recipe.
I	Current place in loop.
INGREDIENT	Cost of ingredient.
SERVINGS	How many to be served this culinary masterpiece you are slaving over.
SERVING	Cost per serving.
COST	Total cost of meal.

The Program

```
100 REM MENU COST PLANNER
110 TRAP 340:REM WATCH FOR MISTAKES
120 PRINT CHR$(125):REM CLEAR SCREEN
130 PRINT
140 PRINT "NUMBER OF INGREDIENTS IN RECIPE"
150 INPUT A
160 REM COME BACK FOR EACH INGREDIENT
170 FOR I=1 TO A
180 PRINT
190 PRINT "COST OF INGREDIENT #";I
200 INPUT INGREDIENT
210 REM ADD COST OF EACH INGREDIENT
220 COST=COST+INGREDIENT
230 NEXT I
240 REM NOW TALLY FIGURES
250 PRINT
260 PRINT "SERVINGS PER RECIPE"
270 INPUT SERVINGS
280 SERVING=COST/SERVINGS
290 PRINT
300 PRINT "TOTAL COST OF MEAL  = $";COST
310 PRINT
320 PRINT "COST PER SERVING    = $";SERVING
330 END
340 TRAP 340:REM RESET MISTAKE TRAP
350 REM START OVER
360 GOTO 140
```

21 / Utility Audit

Have you ever wondered how expensive it is to run the microwave oven for an hour? Or, how about a house full of lights that no one is using? It's an eye-opener to discover that 240 watts worth of light bulbs at 7.5 cents per KWH cost about $4.80 a month, or over $50 a year! Walking family members through this home energy audit can help convince them they really ought to switch off lights in rooms when they leave, or don a warm sweater over that summer T-shirt.

What about the argument that it costs more to switch a lamp off then on again an hour later than to simply let it burn? While that might have been true in the old days when sloppy wall switches let current arc across poor contacts, these days it's simply not true. Turn off lights in empty rooms and save cash.

By the way, it's interesting to note the power supply on the ATARI Home Computer is rated at 20 watts. Monitors and televisions usually are rated between 100 and 500 watts. (Tube sets pull more current than transistorized versions.) How much does it cost to run an ATARI 400 Home Computer for an hour?

To find out, RUN the program. You'll need to know how much you are paying for electricity. Most utility bills list the rate per kilowatt-hour (KWH). Plug in the figure when the program asks for it. (Note: enter 7.5 cents as .075.) Next, plug in the rating, in watts, of the ATARI Home Computer or any other appliance you wish to calculate. After some quick computations,the program will show how much it costs to run that appliance for as many hours as you asked it to tally. Additionally, you'll find out what that cost translates into for a day and for a month.

One last note. A computer doesn't think of ten and half cents (per KWH) as 10.5; instead, it sees it as .105.

How the Program Works

Line 110 lurks in the shadows, waiting for you to hit the wrong key! If you do, the error trap at line 490 will start the program all over again.

Line 120 blanks the screen.

Lines 210 and 220 ask for and input the cost of electricity in your town. We will save this value in RATE and use it as many times as you want to figure different WATTS and HOURS used per day. More on this in a moment.

Lines 240 and 250 input the WATTS burned by whatever appliance you are scrutinizing.

Lines 280 and 290 input how many hours a day this appliance typically hums away at its assigned task.

Line 300 is the workhorse of this program. It computes how much of a kilowatt-hour you are using. Line 320 converts this figure to a two-place decimal, for a neater looking display.

Lines 350 through 440 format and print how much it costs to run the appliance by the hour, day, and month.

Line 470 routes the program back to line 240. This way, if you want to plug in the WATTS of a different appliance, or different running time, you can do so without having to start all over again.

String and Numeric Variables

RATE	Cost per kilowatt-hour (KWH).
WATTS	How many watts used by appliance.
HOURS	Hours used per day.
COST	Cost per hour to use appliance.
DAY	Daily cost based on HOURS per day.
MONTH	Monthly cost based on HOURS times DAY.

The Program

```
100 REM APPLIANCE UTILITY COST CALCULATOR
110 TRAP 490:REM CATCH MISTAKES
120 PRINT CHR$(125):REM CLEAR SCREEN
130 POKE 82,10:REM INDENT LEFT MARGIN
140 PRINT
150 PRINT "UTILITY BILL MONITOR"
160 REM RESET LEFT MARGIN
170 REM RETURN TO NORMAL LEFT MARGIN
180 POKE 82,1
190 PRINT
200  REM GET COST OF ELECTRICITY
210 PRINT "RATE PER KILOWATT HOUR"
220 INPUT RATE
230 PRINT
240 PRINT "WATTS USED BY APPLIANCE"
250 INPUT WATTS
260 PRINT
270 REM GET EXPECTED USE OF APPLIANCE
280 PRINT "HOURS USED PER DAY"
290 INPUT HOURS
300 COST=(WATTS/1000)*RATE
310 REM COMPUTE TO TWO PLACE DECIMAL
320 COST = INT((COST*100)+.5)/100
330 PRINT
340 REM PRINT HOURLY COST
350 PRINT "APPLIANCE COSTS $";COST;"   HOUR"
360 DAY=COST*HOURS:REM NOT NECESSARILY 24 HOUR DAY
370 REM COMPUTE MONTHLY COST
380 MONTH=DAY*30
390 PRINT
400 POKE 85,17:REM SET MARGIN
410 PRINT "$";DAY,"DAY"
420 PRINT
430 POKE 85,17
440 PRINT "$";MONTH,"MONTH"
450 POKE 84,7
460 REM GO BACK FOR FURTHER COMPARISONS
470 GOTO 240
480 REM CATCH MISTAKES
490 TRAP 490
500 PRINT "BAD ENTRY, TRY AGAIN . . .."
510 END
```

22 / Heat Loss Cost Analysis

It has been one of those days. Outdoors it's ten degrees below zero, and a waist-high snowdrift blocks the path to the Detroit-built Belchfire 88 that probably won't start anyway. But somehow, through all the wind and blowing snow, the mailperson delivers the utility bill. Slowly, you open the envelope and read the bad news.

If you want to do something about it, you can. The heat loss program will calculate about how many BTUs slip out through the living room picture window, any other window, or even the old wooden door in the back of the house. Kick in a few other bits of information, and you can tally how much money is lost through poor insulation.

Armed with that kind of data, a homeowner can plug in different heat-saving factors, like installing storm windows, drapes, and a new steel door. The program will indicate how much money you could save by upgrading the house's insulation. Match those savings against the cost of installing the energy conservation items, and you can determine whether or not it would pay to do the work. Most often, it pays off big.

Begin the program by plugging in the efficiency rating of the heat source. It can be any kind of gas or fuel oil furnace, electric baseboard heater, woodstove, or kerosene heater. This efficiency rating is an important figure. It tells how well the unit exchanges its energy for useable heat. For instance, some fuel oil furnaces are rated at 50 percent, and some kerosene heaters are rated at 99 percent efficient.

The next consideration is the BTU value of the fuel. In other words, for each unit expended, be it kilowatt-hour, cord of firewood,

Table 22-1 R-VALUES

Single glass with shade	1.0
Single glass with lined drape	1.3
Double glass or storm window	1.8
Double glass with drape	2.1
Triple glass	2.8
Double glass with shutter	9.6
Wooden door	1.56
Insulated steel door	1.69
Wood siding	.81
Fiberglas insulation (3.5")	11.00
Wall with 3 1/2-in. fiberglas	13.0
Wall with 6-in. fiberglas	22.5

or gallon of fuel oil, how many BTUs will it generate? See the table (next page) to get the appropriate figure.

You'll need to know also how much the unit of energy costs. Enter it as decimal information. For instance, $7\frac{1}{2}$ cents per KWH is .075. $1.30 per gallon would be entered as 1.3, and so on.

Since we are figuring heat loss per year, we need to know how many hours the heat source will be operational per average winter. Each city has a different climate, and thanks to the United States weather service, there is a handy number we can plug in. It's called *degree heating hours* (DHH), and the chart on the next page has a representative number of cities. Check the chart for the city nearest your climatic conditions.

Now to the house. Consider one area at a time. Say we want to know how much heat is lost through a window five feet tall by seven feet wide. Enter each dimension at the prompt. The program will take care of the multiplication needed to get area in square feet.

Next we need to know the R-value, or the area's resistance to heat loss. Again, consult the chart provided. With that final number plugged in, the program will print out:

1. How many BTUs are lost through that surface.
2. How much that loss costs per season.

That information can be shocking. A 5 × 7 foot single-pane window (R1) in a Boston home burning fuel oil with a 60 percent efficient furnace loses 4,725,000 BTUs. All those BTUs, by the way, cost $74 a season.

By adding a storm window and thermal drape, the R-value of the same 5 × 7 foot window is increased from 1 to 2.1. Heat loss drops to 2,249,999 BTUs. Those BTUs, by comparison, cost only $35, or $39 less per year.

Table 22-2 HEAT VALUES

Electricity	3,412 BTUs per KWH
Fuel Oil	138,000 BTUs per gallon
Natural Gas	100,000 BTUs per therm
L. P. Gas	93,000 BTUs per gallon
Firewood (hard)	24,000,000 BTUs per cord
Firewood (soft)	15,000,000 BTUs per cord

To aid in considering insulation upgrades, the program has a built-in feature. After each calculation, the cursor automatically returns to the the R-value input line. Plug in a new R-value, and the new heat loss and dollar figure comes up on screen. It's an easy way to assess whether or not it pays to insulate.

How the Program Works

Line 110 blanks the screen.

Line 120 dimensions BLANK$, leaving room for twenty blank spaces. Line 140 defines BLANK$ as consisting of twenty blank spaces.

Line 160 asks for and inputs the rated efficiency of the heat source. This percentage is stored in the variable EFFICIENCY.

Table 22-3 DEGREE HEATING HOURS

Anchorage, AK	261,000
Flagstaff, AZ	172,000
Denver, CO	151,000
Hartford, CT	148,000
Wilmington, DE	118,000
Boise, ID	139,000
Boston, MA	135,000
Duluth, MN	240,000
Buffalo, NY	168,000
New York, NY	115,000
Portland, OR	111,000
Austin, TX	47,520
Seattle, WA	106,000

Line 180 converts EFFICIENCY to an integer.

Line 210 inputs the BTU value of the fuel.

Line 240 gets the cost of the fuel, be it per gallon, cord, KWH, or whatever.

Line 260 calculates how much heat is lost due to design of the furnace. Line 280 tabulates that cost per million BTUs.

Line 290 converts cost to a two-place decimal, for a nicer looking display.

Line 320 inputs how many hours (DHH) per year the furnace typically will need to run.

Lines 350 through 390 figure the cubic foot area of the window, or door, to be studied.

Line 410 locates BLANK$ where the old R-value, if any, was last printed. Line 420 prints BLANK$ (blank spaces), erasing anything in its path.

Line 430 locates the print statement at line 440 where the BLANK$ was just printed.

Line 470 figures heat loss. This formula is based on area, degree heating hours (DHH), and the R-value.

Lines 510 and 520 locate and print BLANK$ to erase old heat loss in BTUs.

Line 540 prints annual heat loss.

Line 560 converts the loss to a million units. Lines 580 and 590 translate that loss into dollars.

Lines 600 through 630 erase any old entries, and print the new cost.

Line 650 sends us back to try new R-values to see how increasing the R-value helps.

String and Numeric Variables

BLANK$	Twenty blank spaces used to erase old calculations.
EFFICIENCY	Tells how efficiently the heat source uses its fuel.
HEAT	Number of BTUs yielded by one unit of the fuel.
COST	Cost of one unit of fuel.
DHH	Degree heating hours. Number of hours heat source typically will run in winter.

WIDTH	Width of area to be studied.
HEIGHT	Height of area to be studied.
AREA	Surface area.
R	R-value.
LOSS	BTUs (heat) lost through area being studied.
MONEY	How much the heat loss costs per year.

The Program

```
100 REM HEAT LOSS AUDIT
110 PRINT CHR$(125):REM CLEAR SCREEN
120 DIM BLANK$(20):REM WILL USE FOR ERASING
130 REM FILL BETWEEN QUOTES WITH 20 SPACES
140 BLANK$="                    "
150 REM GET EFFICIENCY OF HEATING SOURCE
160 PRINT "EFFICIENCY HEAT SOURCE (PERCENT)";:INPUT EFFICIENCY
170 REM CONVERT TO DECIMAL
180 EFFICIENCY=EFFICIENCY/100
190 PRINT
200 REM FIND OUT BTUs PER GALLON,KWH, ETC
210 PRINT "BTU VALUE OF FUEL";:INPUT HEAT
220 REM HOW MUCH PER GALLON, KWH., ETC
230 PRINT
240 PRINT "COST PER FUEL UNIT";:INPUT COST
250 REM FIGURE COST PER MILLION BTU
260 HEAT=EFFICIENCY*HEAT
270 COST=COST/HEAT
280 COST=COST*1000000
290 COST = INT((COST * 100) + .5)/100
300 PRINT
310 REM GET WEATHER SERVICE INFORMATION
320 PRINT "DEGREE HEATING HOURS";:INPUT DHH
330 PRINT
340 REM GET AREA TO STUDY
350 PRINT "WIDTH OF WINDOW ";:INPUT WIDTH
360 PRINT
370 PRINT "HEIGHT OF WINDOW";:INPUT HEIGHT
380 REM FIGURE AREA OF WINDOW
390 AREA=WIDTH*HEIGHT
400 REM PRINT BLANK STRING OVER OLD R VALUE
410 POSITION 23,13
420 PRINT BLANK$
430 POKE 84,13
440 REM GET NEW R VALUE
450 PRINT "'R' VALUE OF WINDOW ";:INPUT R
460 REM FIGURE HEAT LOSS THROUGH AREA STUDIED
470 LOSS=AREA*DHH*(1/R)
480 REM CONVERT TO INTEGER
490 LOSS = INT((LOSS * 100) + .5)/100
500 REM LOCATE BLANK STRING TO ERASE OLD HEAT LOSS
510 POSITION 23,18
520 PRINT BLANK$:REM ERASE OLD STUFF
530 POKE 84,18:REM PRINT NEW
540 PRINT "HEAT LOSS PER WINTER  ";LOSS;" B.T.U.s"
550 REM CONVERT TO MILLION
```

```
560 LOSS=LOSS/1000000
570 REM FIGURE MONEY LOST PER WINTER
580 MONEY=LOSS*COST
590 MONEY = INT ((MONEY * 100) + .5)/100
600 POSITION 13,20
610 PRINT BLANK$:REM ERASE OLD MONEY
620 POKE 84,20:REM PRINT NEW
630 PRINT "COSTS YOU $";MONEY
640 REM TRY DIFFERENT R VALUE
650 GOTO 400
```

23 / Bulk Purchase Tabulator

Quantity purchases can substantially reduce your food bill. Meat markets, for example, usually sell large cuts at a lower price per pound than the individually wrapped cuts that line the meat case. Those consumer-sized packages are a convenience for which you've been paying extra. Learning a few of the ins and outs of purchasing meat in quantities will save you big money.

"Let the buyer beware" is the watchword. Some unscrupulous meat dealers are noted for butchering the beef and taking out two or three T-bone steaks. Always ask for the fat and bones. It lets the meatcutter know you're not an amateur. He or she will know you expect an accounting for every pound of beef you've paid for. In truth, the butcher could still lift a couple of rib-eyes, substitute another customer's bones, and you'd never know. But the idea is to come across with a little more savvy than the average person.

Bait-and-switch is another trick to watch out for. Say you've spent months learning how to tell the kind of marbling that insures a tender portion. You pick out what looks like a good cut, and that's the last you ever see of it as it's wheeled into the back room for processing. Months later, you shake your head and wonder how a selection that looked so tender could be so tough. What happened? Well, you did see a good portion, which ten other people probably picked for their own that day. And you ended up with an entirely different piece of meat. Because of this bait-and-switch practice, some zealous customers demand to watch the carving-up practice.

Don't be surprised when a full 30 to 40 percent of the original weight disappears into the heap of bones and fat. That's standard, and you'll pay for the waste at the same per-pound cost as the ribs, roast, and steaks. Remember, you are buying the entire portion, not just the good parts.

Perhaps you'd like to try your hand at cutting the meat yourself. If you're interested, the Government Printing Office puts out some very informative literature, amounting to a free crash-course in butchering meat. Ask for pamphlet AFS 6-4 entitled "How to Save Money with Large Cuts of Meat."

No matter who does the bloody work, storage is an important factor for a quarter of a ton of beef. How will you store it: in the top compartment of the refrigerator-freezer, a commercial locker, or a home freezer? For purposes of calculation, consider that one cubic foot of freezer space will store about thirty-five pounds of wrapped meat. If the shapes of the cuts are irregular, figure on fewer pounds per cubic foot.

Wrapping technique is most important. Moisture loss is bad for meat and tends to occur with exposure to air. To prevent it, use heavy aluminum foil, heavily waxed freezer paper, or laminated paper. Plastic bags will work well if double-bagged, twisted shut, doubled back on the twist, and then sealed with wire ties.

When two or more steaks are wrapped in the same pack, slide a couple sheets of waxed paper in between the two to prevent them from freezing together. Once the individual cuts are wrapped, mark the contents and the date.

How long will frozen meats stay good? That depends on the temperature. The colder the temperature, the longer meat will keep. At zero degrees, steaks and roasts can keep from eight months to a year. Ground beef and lamb are good for about four months; while ground pork keeps from one to three months.

While this program helps figure the cost effectiveness of buying a quarter or side of beef, it could easily be modified to consider bushels of tomatoes, a peck of potatoes, or even big bunches of carrots for canning.

Since the program is written for meat, tell the computer how many pounds you bought, at what price per pound, how much the butchering fee was, and, finally, how many pounds were left after processing.

After the last entry, the program will tell how much you paid for the total package, and how much per pound it effectively cost.

How the Program Works

Line 110 blanks the screen.

Line 130 inputs the pounds of whatever food is being considered. Maybe it's carrots or cabbage about to be canned.

Line 170 calculates how much it costs to buy that food.

Line 190 inputs the processing charge. For meat, it means butchering. For carrots on their way to being canned, processing includes salt and mason jars.

Line 230 considers the reality that not all the food is useable. There is always waste. After processing, how many pounds are left?

Line 240 adds the original purchase price to the cost of any processing or supplies.

Line 250 figures out the cost PERPOUND; while line 260 converts the figure to a two-place decimal.

String and Numeric Variables

BEEF	Pounds of food bought.
POUND	Price per pound.
PROCESSING	Cost of butchering, canning, and so on.
BEEFLEFT	Amount of food left after any waste is subtracted.
PRICE	The sum of food purchase, plus processing and supplies.
PERPOUND	Ultimate cost per pound to make this bulk purchase.

The Program

```
100 REM SIDE OF BEEF COST ADVANTAGE
110 PRINT CHR$(125):REM CLEAR SCREEN
120 PRINT "POUNDS OF BEEF PURCHASED"
130 INPUT BEEF
140 PRINT
150 PRINT "PRICE PER POUND"
160 INPUT POUND
170 PRICE=BEEF*POUND
180 PRINT
190 PRINT "PROCESSING CHARGE"
200 INPUT PROCESSING
210 PRINT
220 PRINT "POUNDS OF BEEF AFTER BUTCHERING"
230 INPUT BEEFLEFT
240 PRICE=PRICE+PROCESSING
250 PERPOUND=(PRICE/BEEFLEFT)
260 PERPOUND = INT((PERPOUND * 100)+ .5)/100
270 PRINT "-----------------------"
280 PRINT "TOTAL","$";PRICE
290 PRINT :PRINT
300 PRINT "AVERAGE","$";" PERPOUND "
```

24 / Smart Typewriter— Dumb Word Processor

A word processor doesn't have to be loaded with frills to be valuable, as long as you don't expect right-margin justification, pagination, and all the fancy extras that come with a commercial piece of software. Without the frills, this program will perform almost all of your text-writing needs. Write letters, poetry, a best-selling novel, whatever you'd like. In fact, most of this book was written with this program. Here's how to use it.

Type in the program, SAVE it, then RUN. The menu displays three choices. Just press 1, 2, or 3. There's no need to hit RETURN. If your choice was 1, the screen will blank and ask for the name of the file under which you want to save your work. With a disk drive, the usual filename restrictions apply. If you use a cassette recorder, enter C:. Here's a neat feature if you want to print your work as you go. Instead of C: or D:FILENAME, enter P:, for printer. If you do, every time the buffer fills up with about 128 characters, those characters will be printed, just as you typed them.

Back to the filename selection. Once you enter D, C, or P, the screen will blank and a prompt will tell you it's okay to start typing. If you make a mistake, use the backspace key to correct it.

Each line of text can hold up to 255 characters. A warning beep, however, will let you know once you've typed thirty of them. If you detest that little noise, delete the appropriate program line to eliminate it.

This is important. Once you have typed a line and hit RETURN, it can't be changed. It is as good as written in stone (silicon?). You can't correct it. Also, the break key has been disabled because of its close proximity to the backspace key, the logic being the last

thing your sunny disposition needs is to hit the break key instead of the backspace key in the middle of composing an important letter!

When you finish drafting the document and are ready to quit, hit RETURN at the beginning of the line. It must be the first, and only, character of the line. With this, the write-text program ends, and control returns to the task menu. Remember, the break key is disabled, so the only way to quit is by hitting RETURN.

So now we're back at the menu. Choose 1, 2, or 3. Let's say that you've typed a 2 because you'd like to read over a letter you've written to your Congressman regarding an issue very sensitive to you. The screen will clear and ask for the filename of the all-important document. This is a good time to talk about choice of filenames. Sometimes it's hard to remember how a document was filed. It helps to have a system. For instance, label all correspondence with the last name of the recipient, followed by an extender labeling it as a letter. One example might be KENNEDY.LET. A speech, on the other hand, might be labeled: WHALES:TXT. It's not terribly important which system you use, as long as it works.

Once you've told the ATARI Home Computer which file you want to call up, the screen will clear and display said file's first twenty lines. Then it stops. To see twenty more lines, tap any key, except break—remember it has been disabled. Once we've read the last line of the file, a little message will let us know. Again, hit any key to go on.

After having read the length of the file, control will loop back to the task menu. If you're finished work, type a 3 and the program will end.

How the Program Works

Line 130 opens the keyboard. Line 140 blanks the screen.

Lines 160 through 250 display the task menu.

Lines 280 through 300 return the user's choice and send the program off to do its chosen task.

Line 310 asks for a new choice if the user has typed any number other than a 1, 2, or 3.

Lines 320 through 430 assign the filename where the work will be stored.

Line 450 looks for the letter C, which is the clue you're using a cassette recorder. If so, the program will initialize a cassette file before writing to it.

Line 470 opens the file.

Lines 490 through 520 disable the break key.

Line 610 dumps any old characters from the keyboard buffer. This prevents mistakes each time the computer halts to send a line to the file.

Line 620 reads each character as you type it.

Line 640 looks for a carriage return. If it's the first character in the line, it means you've finished work.

Line 660 counts the number of characters typed per line. Line 680 looks for the backspace character.

Line 710 prints a beep-beep if you've typed 30 characters on the line. Delete this line if the sound annoys you.

Line 740 adds each character typed into one big long line, a string variable that holds up to 255 characters. A carriage return signals the end of that line.

Line 780 sends each new line to the file. Line 790 empties the string. Line 800 goes to get the next line of text.

Line 820 closes the file. Line 840 resets the keyboard to uppercase only. This is a safeguard. If you ever forgot to reset it manually after writing, and then tried to enter a request to read or write any file, the program would crash. Example: "D:filename" won't work, while "D:FILENAME" will.

Line 860 sends us back to the task menu.

Line 870 blanks the screen. Lines 880 through 950 display the prompts that ask which file you want to call up. Line 980 opens that file.

Line 1000 tells the computer to look for the end of the file. When it reads that far, then go to line 1240, where we'll wait for any key to be pressed.

Line 1020 checks to see if it is a cassette file. If so, it dumps the dummy data written when the file was initialized.

Line 1040 reads one line of text at a time. Line 1050 counts the lines displayed on the screen. Line 1090 stops the display every twenty lines by jumping down to line 1120 and waiting for you to press a key.

Lines 1120 through 1220 wait for any key to be pressed before showing any more text. Lines 1240 through 1330 close the file, and wait for any key to be pressed.

Lines 1350 through 1390 are the finished routine.

Lines 1410 through 1610 initialize a new cassette file by writing 128 bytes of dummy data.

String and Numeric Variables

FILENAME$	The cassette or disk file to be opened.
LINE$	Line of text being written or read to or from FILENAME$.
CHOICE	Which menu number (job) user wants performed.
FLAG	Was last character typed a RETURN?
X	Returns status of break key memory locations.
KEY	Has any key been pressed?
COUNT	Number of lines read and displayed from file.

The Program

```
100 REM SMART TYPEWRITER/DUMB WORD PROCESSOR
110 DIM FILENAME$(20)
120 DIM LINE$(255)
130 OPEN #1,4,0,"K:"
140 PRINT CHR$(125):REM BLANK SCREEN
150 REM DISPLAY TASK MENU
160 POSITION 12,2
170 PRINT "  W.P. TASK CHART "
180 POSITION 2,5
190 PRINT "1. WRITE DOCUMENT"
200 POSITION 2,7
210 PRINT "2. READ DOCUMENT"
220 POSITION 2,9
230 PRINT "3. FINISHED"
240 POSITION 5,11
250 PRINT "PICK 1 - 3 ";
260 REM OPEN KEYBOARD FOR READ
270 REM GET CHOICE
280 GET #1,CHOICE
290 REM GO DO APPROPRIATE TASK
300 ON CHOICE-48 GOTO 320,870,1360
310 GOTO 140
320 PRINT CHR$(125)
330 REM DISPLAY FILE NAMING SEQUENCE
340 POSITION 12,2
350 PRINT " WRITE DOCUMENT "
360 POSITION 9,7
370 PRINT " EXAMPLE : D:LETTER.TXT"
380 POSITION 20,8
390 PRINT "C: (CASSETTE USERS)"
400 POSITION 20,9
410 PRINT "P: (PRINTER)"
420 POSITION 2,5
430 PRINT "SAVE DOCUMENT AS ";:INPUT FILENAME$
440 REM IF CASSETTE FILE GO INITIALIZE
450 IF FILENAME$(1,1)="C" THEN GOTO 1410
460 REM OPEN TEXT FILE OR PRINT TEXT
```

```
470 OPEN #2,8,0,FILENAME$
480 REM ** DISABLE THE BREAK KEY **
490 X=PEEK(16)
500 IF PEEK(16)<128 THEN 530
510 POKE 16,X-128
520 POKE 53744,X-128
530 PRINT CHR$(125)
540 POSITION 7,1
550 PRINT " PLEASE BEGIN TYPING "
560 PRINT
570 FLAG=0:REM CLEAR END OF LINE FLAG
580 LINE$="":REM DUMP STRING CONTENTS
590 COUNT=0
600 REM GET ONE CHARACTER AT A TIME
610 POKE 764,255
620 GET #1,KEY
630 REM WATCH FOR CARRIAGE RETURN
640 IF COUNT=0 AND KEY=155 THEN 820
650 PRINT CHR$(KEY);
660 COUNT=COUNT+1
670 REM WATCH FOR BACKSPACE
680 IF KEY=126 THEN COUNT=COUNT-2
690 IF COUNT<1 THEN COUNT=1
700 REM 30 CHARACTERS THEN BEEP
710 IF COUNT=30 THEN PRINT CHR$(253);
720 IF COUNT=37 THEN KEY=155
730 IF KEY=155 THEN FLAG=1
740 LINE$(LEN(LINE$)+1)=CHR$(KEY)
750 IF FLAG=1 THEN GOTO 780
760 GOTO 610
770 REM GET AND PRINT TEXT LINES
780 PRINT #2;LINE$;
790 LINE$=""
800 GOTO 570
810 REM CLOSE THE TEXT FILE
820 CLOSE #2
830 REM RESTORE UPPER CASE IF SET TO LOWER
840 POKE 702,64
850 REM GO TO TASK MENU
860 GOTO 140
870 PRINT CHR$(125):REM BLANK THE SCREEN
880 REM DISPLAY FILE READING SEQUENCE
890 POSITION 12,2
900 PRINT " READ DOCUMENT "
910 POSITION 10,7
920 PRINT " EXAMPLE : D:LETTER.TXT"
930 POSITION 21,8
940 PRINT "C: (CASSETTE USERS)"
950 POSITION 2,5
960 PRINT "SAVED DOCUMENT AS ";:INPUT FILENAME$
970 REM OPEN TEXT FILE FOR READ
980 OPEN #2,4,0,FILENAME$
990 PRINT CHR$(125)
1000 TRAP 1240:REM WATCH FOR END OF FILE
1010 REM IF CASSETTE FILE DUMP DUMMY DATA
1020 IF FILENAME$(1,1)="C" THEN INPUT #2;LINE$
1030 REM READ A LINE
1040 INPUT #2,LINE$
1050 COUNT=COUNT+1
1060 REM PRINT THE LINE
1070 PRINT LINE$
1080 REM 20 LINES PRINTED?
1090 IF COUNT=20 THEN GOSUB 1120
1100 GOTO 1040
```

```
1110 REM WAIT ROUTINE
1120 PRINT :PRINT
1130 POSITION 12,22
1140 PRINT " HIT ANY KEY ";
1150 REM WAIT FOR A KEY TO BE PRESSED
1160 POKE 764,255
1170 IF PEEK(764)=255 THEN 1170
1180 POKE 764,255
1190 PRINT :PRINT
1200 REM ZERO OUT THE CHARACTER COUNT
1210 COUNT=0
1220 RETURN
1230 REM END OF FILE ROUTINE
1240 PRINT :PRINT
1250 POSITION 12,22
1260 PRINT " END OF FILE ";
1270 POKE 764,255
1280 IF PEEK(764)=255 THEN 1280
1290 POKE 764,255
1300 PRINT :PRINT
1310 COUNT=0
1320 CLOSE #2
1330 GOTO 140
1340 REM FINISHED ROUTINE
1350 CLOSE #1
1360 POSITION 2,21
1370 PRINT "GOOD BYE!"
1380 POSITION 2,22
1390 END
1400 REM CASSETTE FILE UTILITY (INITIALIZE)
1410 PRINT CHR$(125):REM BLANK THE SCREEN
1420 REM PRINT INSTRUCTIONS
1430 POSITION 8,2
1440 PRINT " INITIALIZE CASSETTE FILE "
1450 POSITION 5,5
1460 PRINT "1. INSERT TAPE"
1470 POSITION 5,7
1480 PRINT "2. PRESS RECORD/PLAY"
1490 POSITION 5,9
1500 PRINT "3. HIT RETURN "
1510 REM PRINT WAIT MESSAGE
1520 POSITION 5,11
1530 PRINT "4. WAIT . . .";
1540 REM OPEN THE CASSETTE CHANNEL (WRITE)
1550 OPEN #2,8,0,"C:"
1560 REM WRITE DUMMY DATA (R. = REM)
1570 FOR COUNT=1 TO 127
1580 PRINT #2;CHR$(32);
1590 NEXT COUNT
1600 PRINT #2
1610 GOTO 490
```

25 / Carpool Worksheet

It's no surprise that carpooling saves money, whether it's a daily trip from the suburbs into work, or a group of friends sharing the expenses on a special trip. Regardless of the motivation, this program makes divvying up the costs a more pleasant task.

This is a very uncomplicated system. Simply plug in the round-trip miles, the mileage of the vehicle you've be traveling in, along with the cost per gallon of fuel, how many folks are going, plus any bridge or tollway fares you may be paying.

That done, the results pop out. Perhaps you didn't like the results—too expensive. Okay, leave the big factory Lincoln in Ralph's driveway and take Mary's little red Chrysler. See how much difference the mileage makes.

How the Program Works

Line 150 blanks the screen. Lines 160 through 310 format the display that asks for all the particulars, such as mileage, miles driven, cost of fuel, and so on

Lines 330 through 410 calculate the costs. Notice lines 390, 400, and 410. They factor each of their variables down to a two-place decimal. That prevents a messy answer like:

```
COST PER MILE = .04567891234
```

Lines 430 through 470 display the calculations. Line 490 sends us back to get new data; that is, if you're interested in comparing different vehicles and their mileage.

String and Numeric Variables

MILES	Miles (round trip) to be driven.
MPG	Mileage per gallon.
FUEL	Cost of gas or diesel fuel.
NUMBER	Passengers sharing expenses.
TOLLS	Total amount of bridge and highway fares.
PERMILE	Cost per round-trip mile.
RIDER	Cost each passenger.
TRIP	Total cost of trip.

The Program

```
100 REM ****************************
110 REM *                          *
120 REM *   CAR POOL WORKSHEET     *
130 REM *                          *
140 REM ****************************
150 PRINT CHR$(125):REM BLANK THE SCREEN
160 POSITION 5,2
170 PRINT " CAR POOL EXPENSE SHEET "
180 REM DETAIL THE EXPENSES
190 POSITION 2,5
200 PRINT "1. ROUND-TRIP MILES PER DAY ";:INPUT MILES
210 POSITION 2,7
220 PRINT "2. CAR'S M.P.G. (HIGHWAY) ";:INPUT MPG
230 POSITION 2,9
240 PRINT "3. COST FUEL PER GAL.";:INPUT FUEL
250 POSITION 2,11
260 PRINT "3. NUMBER OF RIDERS ";:INPUT NUMBER
270 POSITION 2,13
280 PRINT "4. TOLLWAY & BRIDGE FARES";:INPUT TOLLS
290 PRINT
300 PRINT "-----------------------------------"
310 PRINT
320 REM CALCULATE COSTS
330 GALLONS=MILES/MPG
340 FUELCOST=GALLONS*FUEL
350 TRIP=TOLLS+FUELCOST
360 PERMILE=TRIP/MILES
370 RIDER=TRIP/NUMBER
380 REM CONVERT TO TWO DECIMAL PLACE
390 PERMILE=INT((PERMILE*100)+0.5)/100
400 RIDER=INT((RIDER*100)+0.5)/100
```

```
410 TRIP=INT((TRIP*100)+0.5)/100
420 REM DISPLAY THE TABULATIONS
430 PRINT "COST PER MILE = $";PERMILE
440 PRINT
450 PRINT "COST PER RIDER  $";RIDER
460 PRINT
470 PRINT "COST FOR TRIP   $";TRIP
480 REM FIGURE WITH NEW VARIABLES
490 GOTO 190
```

26 / Music Composer

The ATARI Home Computer is a maestro when it comes to making music. With this composer, one can easily mimic a synthesizer keyboard boasting three octaves with low C, middle C, and high C. Moreover, each octave has its own voice. In this case, it means that the last note played, within its respective octave, will echo until either a new note in that octave is played, or the voice is "quieted" by means of its assigned function key. These are the bottom three yellow keys at the right of the console.

With the Music Composer program up and runing, one by one press keys A through K. This plays middle C, from *do* to *do*. Now press any one of those keys while holding down the yellow SELECT button. The echo will become silent.

Likewise, play the notes Q through I to get a feel for high C. Only, this time, play a note and hold down the yellow OPTION button to silence the voice.

Similarly, you can use the Z through comma keys to play the low C octave, and dampen its sound with the yellow START key.

Improvise your own tunes; let your imagination be your guide. Perhaps you will be delivered to a Gothic cathedral, a cobwebbed castle, or a rock concert sound stage.

How the Program Works

Lines 170 through 220 paint the screen and display the program's logo. When typing in line 220, enter the uppercase letters

M and C as usual, but press the inverse key for the remaining lowercase characters. In Graphics 2 mode you can display one color of characters by using the standard character set, and another by using the inverse. Experiment with upper- and lowercase. Each one paints different colors.

Line 250 gets each key's number as you press it. Lines 280 through 530 convert that number to represent a musical note. Lines 550 through 570 separate those notes into the three different octaves.

Lines 590 through 610 play those notes. Line 590 can be modified for an interesting variation. Change it to read:

```
590 FOR VOLUME = 15 TO 1 STEP -1
```

Lines 630 through 650 silence the octaves, by fingertip command.

String and Numeric Variables

KEY	Key pressed.
PITCH	Musical note to be played.
VOICE	One of three voices being played.
VOLUME	Loudness.

The Program

```
100 REM ****************************
110 REM *                          *
120 REM *                          *
130 REM *  THE MUSIC COMPOSER      *
140 REM *                          *
150 REM *                          *
160 REM ****************************
170 REM CHOOSE GRAPHICS MODE
180 GRAPHICS 2+16
190 REM PAINT THE SCREEN
200 SETCOLOR 4,8,4
210 POSITION 2,5
220 PRINT #6;"Music Composer"
230 OPEN #1,4,0,"K:"
240 REM KEYBOARD SUPPLIES MUSIC
250 GET #1,KEY
260 REM CONVERT TO MUSICAL NOTE
270 REM Z TO COMMA ARE LOW C
280 IF KEY=90 THEN PITCH=242:REM Z
```

```
290 IF KEY=88 THEN PITCH=217:REM X
300 IF KEY=67 THEN PITCH=193:REM C
310 IF KEY=86 THEN PITCH=182:REM V
320 IF KEY=66 THEN PITCH=162:REM B
330 IF KEY=78 THEN PITCH=144:REM N
340 IF KEY=77 THEN PITCH=128:REM M
350 IF KEY=44 THEN PITCH=121:REM ,
360 REM A TO K ARE MIDDLE C
370 IF KEY=65 THEN PITCH=120:REM A
380 IF KEY=83 THEN PITCH=108:REM S
390 IF KEY=68 THEN PITCH=96:REM D
400 IF KEY=70 THEN PITCH=91:REM F
410 IF KEY=71 THEN PITCH=81:REM G
420 IF KEY=72 THEN PITCH=72:REM H
430 IF KEY=74 THEN PITCH=64:REM J
440 IF KEY=75 THEN PITCH=60:REM K
450 REM Q TO I ARE HIGH C
460 IF KEY=81 THEN PITCH=59:REM Q
470 IF KEY=87 THEN PITCH=53:REM W
480 IF KEY=69 THEN PITCH=47:REM E
490 IF KEY=82 THEN PITCH=45:REM R
500 IF KEY=84 THEN PITCH=40:REM T
510 IF KEY=89 THEN PITCH=35:REM Y
520 IF KEY=85 THEN PITCH=31:REM U
530 IF KEY=73 THEN PITCH=29:REM I
540 REM EACH OCTAVE HAS ITS OWN VOICE
550 IF PITCH>59 THEN IF PITCH<121 THEN VOICE=1:GOTO 590
560 IF PITCH>120 THEN VOICE=2:GOTO 590
570 IF VOICE<60 THEN VOICE=0
580 REM PLAY EACH NOTE AND ITS VOICE
590 FOR VOLUME=15 TO 10 STEP -1
600 SOUND VOICE,PITCH,14,VOLUME
610 NEXT VOLUME
620 REM CONSOLE BUTTONS SILENCE VOICE
630 IF PEEK(53279)=3 THEN SOUND 0,0,0,0:REM OPTION
640 IF PEEK(53279)=5 THEN SOUND 1,0,0,0:REM SELECT
650 IF PEEK(53279)=6 THEN SOUND 2,0,0,0:REM START
660 REM GET NEXT NOTE
670 GOTO 250
```

27 / Typing Tutor

If you use a personal computer, efficient typing is a must, to avoid the frustration of long hours at the keyboard, hunting and pecking, writing and correcting. If you understand the principles of touch typing, you are ahead of the game.

Briefly, the system requires you to look at the CRT or TV screen; it's forbidden to look at the keys. This is where the speed comes in. Train the mind so that the fingers instinctively know where the A, G, or L is located. If the mind and eye committee isn't slowed down by the hunt-and-peck variety of typing, it will be able to move through its creative tasks much more quickly.

This takes practice. The typing tutor provides an interesting way to attain that practice and test yourself. RUN the program and see the single character displayed in the center of the screen. Your job is to simply type each one as it comes up. If you respond correctly, a new one is displayed. However, if you hit the wrong key, a blunder sound is emitted from the TV speaker. Never fear, there are no penalties for being wrong; after all, you'll do better next time.

After a few workouts with the tutor, word processing can become much less tedious, and programming will likewise be much less frustrating. When hard at it, remember to keep your eyes on the screen, and let your fingers do the typing.

How the Program Works

Lines 220 and 240 set up the screen to display those big Graphics 2 letters.

Line 260 opens the keyboard to read your responses as you are tested.

Line 280 is the program's heart and soul. It generates a random number between 1 and 26. Each number corresponds to a letter in the alphabet, from A to Z.

Line 300 adds 65 to each one of those numbers to convert it to ATASCII, or the set of numbers the computer understands for each A, B, C, or D.

Lines 320 and 330 display the letter you're being tested on. Line 350 gets your response. Line 410 compares your response with the correct answer. If you make a mistake, you'll hear about it.

Line 430 goes back to the well to test you on another character.

String and Numeric Variables

TYPE	A random number representing a letter from A to Z.
KEY	The letter you typed.

The Program

```
100 REM ****************************
110 REM *                          *
120 REM *                          *
130 REM *     QWERT  YUIOP         *
140 REM *                          *
150 REM *          THE             *
160 REM *                          *
170 REM *     TYPING TUTOR         *
180 REM *                          *
190 REM *                          *
200 REM ****************************
210 REM GR.2 FOR BIG CHARACTERS
220 GRAPHICS 2+16
230 REM PAINT THE SCREEN
240 SETCOLOR 4,8,4
250 REM OPEN KEYBOARD FOR READ
260 OPEN #1,4,0,"K:"
270 REM PICK NUMBER BETWEEN 1 AND 26
280 TYPE=INT(26*RND(1))
290 REM CONVERT TO LETTER OF ALPHABET
300 TYPE=TYPE+65
310 REM DISPLAY RANDOM LETTER
320 POSITION 9,5
330 PRINT #6;CHR$(TYPE);
340 REM GET LETTER STUDENT TYPED
350 GET #1,KEY
360 REM PRINT LETTER STUDENT TYPED
370 POSITION 9,5
380 PRINT #6;CHR$(KEY);
```

```
390 REM COMPARE THE TWO
400 REM IF MISTAKE MADE, MAKE SOUND
410 IF KEY<>TYPE THEN SOUND 0,99,10,15:FOR I=1 TO 200:NEXT I
420 SOUND 0,0,0,0
430 REM TEST THE NEXT LETTER
440 GOTO 280
```

28 / Home Inventory Log

A home inventory log is a good idea. Should the misfortune of burglary, hurricane, or fire ever befall you, it becomes much easier for the police department to reunite you with your possessions, or for an insurance company to accurately process your claim.

But what if you use a computer to record model numbers and serial numbers and the diskette burns up in the fire? Do you have a fireproof safe in the study? It may not help. Cassette and diskette files may melt. So it's a good idea to backup that information on paper as well. Printer paper has a flash point of about 1400 degrees and will survive long after a tape has melted into a big lump of goo. It would be even better to store your log in an inexpensive safety deposit box where you bank.

To use the program, type the number 1, 2, or 3. The number 1 will call up the part of the program that will allow logging each possession, item by item. Fill in a descriptive name, the purchase price, and serial number. When you're done, hit just the return key when it asks for the item's name. That closes the file you've just written, and sends you back to the menu.

Type the number 2, and you can read the inventory log.

If you've made some new purchases, and want to add them to the list, type in number 1, and the information will be added to the bottom of the old file.

If you've finished working with the program, a 3 will end it.

How the Program Works

Line 120 opens the keyboard to help in the menu selection. Lines 140 through 170 dimension the variables we'll be using to send information from the keyboard to the inventory log.

Lines 180 through 270 format and display the task menu. Line 280 gets the user's choice, and line 290 sends the program off to do its work.

Lines 310 through 320 ask for the name of this inventory log and open it. Line 310 is an error trap. If the file does not exist, as would be the case the first time you write to the log, lines 810 through 840 create a new file.

Line 440 blanks the screen. Lines 460 through 510 display the header, or title page, if you will.

Lines 520 through 610 format and display the prompt asking for each item to be recorded. But there's something special going on here. Notice line 560 and its PRINT CHR$(254). 254 is ATASCII for "delete character." That means every time a PRINT CHR$(254) occurs, a character will disappear. In order to erase the old item's name when we wish to insert a new one, we erase the old one by printing "delete character," as many times as the old item was characters long.

Line 610 watches for the user to enter a carriage return as the only response to an item request. This done, the file is closed and control loops back to the task menu.

Lines 630 through 750 use the same scenario—to first input, then erase the information we need.

Line 770 writes the item, its price, and serial number to the inventory file.

Lines 910 through 990 ask for the name of the file to be read. By the way, it dosen't have to be an inventory file. You can read just as easily a BASIC program you've written.

Line 1050 checks to see if it's a cassette file. If so, the dummy information written when it was initialized is simply disregarded.

Lines 1070 through 1120 read and display twenty lines at a time. Lines 1140 through 1220 wait for any key to be pressed before allowing a read of another twenty lines. This is a big help when there's a lot of information to plow through. Without this subroutine, the text would flash past faster than even a speed reader could absorb.

When the program opened this file, line 1030 set an error trap. When a program reads to the bottom of a file, an ERROR 136 occurs. The error trap then shoots the program down to line 1240. Hit any key, and the program will loop back to the task menu.

Lines 1320 through 1520 initialize a cassette file by writing dummy bytes to it.

String and Numeric Variables

KEY	Tells which job you want done.
FILENAME$	The name of the file to be opened.
ITEM$	Name of the item being recorded.
PRICE$	Price paid for item.
SERIAL$	Serial number of item being recorded.
COUNT	Number of lines read and displayed from inventory file.

The Program

```
100 REM HOME INVENTORY LOG
110 REM OPEN KEYBOARD FOR READ
120 OPEN #1,4,0,"K:"
130 REM DIMENSION VARIABLES
140 DIM FILENAME$(20)
150 DIM ITEM$(39)
160 DIM PRICE$(39)
170 DIM SERIAL$(39)
180 PRINT CHR$(125):REM BLANK SCREEN
190 REM DISPLAY TASK MENU
200 POSITION 10,2
210 PRINT " HOME INVENTORY LOG "
220 POSITION 2,5
230 PRINT "1. LOG POSSESSIONS"
240 POSITION 2,7
250 PRINT "2. READ INVENTORY"
260 POSITION 2,9
270 PRINT "3. QUIT ";
280 GET #1,KEY
290 ON KEY-48 GOTO 320,910,860
300 GOTO 180
310 TRAP 810
320 PRINT CHR$(125):REM BLANK THE SCREEN
330 POSITION 12,2
340 PRINT " RECORD POSSESSIONS "
350 POSITION 7,7
```

```
360 PRINT " EXAMPLE :  D:INVENT.ORY"
370 POSITION 19,9
380 PRINT "C: (CASSETTE USERS)"
390 POSITION 2,5
400 PRINT "FILE SAVE(D) AS ";:INPUT FILENAME$
410 IF FILENAME$(1,1)="C" THEN 1320
420 TRAP 810
430 OPEN #2,9,0,FILENAME$
440 PRINT CHR$(125):REM BLANK THE SCREEN
450 REM PRINT PAGE HEADER
460 POSITION 9,1
470 PRINT "----------------"
480 POSITION 10,2
490 PRINT "HOME INVENTORY"
500 POSITION 9,3
510 PRINT "----------------"
520 POSITION 2,5
530 PRINT "    ITEM        "
540 POSITION 2,6
550 REM ERASE ANY OLD CHARACTERS
560 FOR I=1 TO LEN(ITEM$):PRINT CHR$(254);:NEXT I
570 POSITION 2,6
580 REM GET THE ITEM NAME
590 INPUT ITEM$
600 REM CARRIAGE RETURN ONLY WHEN DONE
610 IF ITEM$="" THEN CLOSE #2:GOTO 180
620 POSITION 2,8
630 PRINT "  PURCHASE PRICE "
640 POSITION 2,9
650 REM ERASE ANY OLD CHARACTERS
660 FOR I=1 TO LEN(PRICE$):PRINT CHR$(254);:NEXT I
670 POSITION 2,9
680 INPUT PRICE$
690 POSITION 2,11
700 PRINT "  SERIAL NUMBER "
710 POSITION 2,12
720 REM ERASE ANY OLD CHARACTERS
730 FOR I=1 TO LEN(SERIAL$):PRINT CHR$(254);:NEXT I
740 POSITION 2,12
750 INPUT SERIAL$
760 REM SAVE THE INFORMATION IN FILE
770 PRINT #2;ITEM$,PRICE$,SERIAL$
780 REM GET NEXT ENTRY
790 GOTO 520
800 REM CREATE FILE (FIRST TIME ONLY)
810 CLOSE #2
820 OPEN #2,8,0,FILENAME$
830 CLOSE #2
840 GOTO 430
850 REM QUIT ROUTINE
860 CLOSE #1
870 POSITION 16,19
880 PRINT "BYE!"
890 POSITION 2,21
900 END
910 PRINT CHR$(125):REM BLANK THE SCREEN
920 POSITION 8,2
930 PRINT " SEE HOME INVENTORY "
940 POSITION 5,7
950 PRINT " EXAMPLE :  D:INVENT.ORY"
960 POSITION 17,9
970 PRINT "C: (CASSETTE USERS)"
980 POSITION 2,5
990 PRINT "FILE SAVED AS ";:INPUT FILENAME$
```

```
1000 OPEN #2,4,0,FILENAME$
1010 PRINT CHR$(125):REM BLANK THE SCREEN
1020 REM WATCH FOR END OF FILE
1030 TRAP 1240
1040 REM IF CASSETTE FILE DUMP DUMMY INFO.
1050 IF FILENAME$(1,1)="C" THEN INPUT #2;ITEM$
1060 REM GET FIRST LINE FROM FILE
1070 INPUT #2;ITEM$
1080 REM COUNT LINES PRINTED
1090 COUNT=COUNT+1
1100 PRINT ITEM$
1110 IF COUNT=20 THEN GOSUB 1140
1120 GOTO 1070
1130 REM PRINT 20 LINES AT A TIME
1140 PRINT
1150 PRINT
1160 POSITION 12,22
1170 PRINT " HIT ANY KEY ";
1180 GET #1,KEY
1190 COUNT=0
1200 PRINT
1210 PRINT
1220 GOTO 1070
1230 REM CLOSE WHEN DONE READING
1240 CLOSE #2
1250 PRINT
1260 PRINT
1270 POSITION 14,22
1280 PRINT " END OF FILE";
1290 GET #1,KEY
1300 GOTO 180
1310 REM CASSETTE FILE UTILITY (INITIALIZE)
1320 PRINT CHR$(125):REM BLANK THE SCREEN
1330 REM PRINT INSTRUCTIONS
1340 POSITION 8,2
1350 PRINT " INITIALIZE CASSETTE FILE "
1360 POSITION 5,5
1370 PRINT "1. INSERT TAPE"
1380 POSITION 5,7
1390 PRINT "2. PRESS RECORD/PLAY"
1400 POSITION 5,9
1410 PRINT "3. HIT RETURN "
1420 REM PRINT WAIT MESSAGE
1430 POSITION 5,11
1440 PRINT "4. WAIT . . .";
1450 REM OPEN THE CASSETTE CHANNEL (WRITE)
1460 OPEN #2,8,0,"C:"
1470 REM WRITE DUMMY DATA (R. = REM)
1480 FOR COUNT=1 TO 127
1490 PRINT #2;CHR$(32);
1500 NEXT COUNT
1510 PRINT #2
1520 GOTO 440
```

29 / Tax Deduction Recorder

Organization yields better results in almost any venture. And when the final consideration is whether or not you'll be able to declare all your justified deductions, that method of organization becomes an important aspect of tax records.

This system is simple. Once every day, week, or month—or at the conclusion of a business trip—sit down with all your receipts. Sort them in chronological order; then enter each one. Log each kind of deduction—entertainment, travel, meals, whatever. Also record the item purchased—lunch, newspaper, or financial report. Next enter the dollar amount, followed by the date.

With the receipts gathered into this expense ledger, yearly tax calculations will be smoother.

How the Program Works

Lines 110 through 160 dimension and declare the string variables. Line 180 opens the keyboard for a read.

Lines 190 through 300 format and display the task menu. Line 320 gets the user's choice of tasks, and line 330 sends the program off to perform that task. If the user entered any number other than a 1, 2, or 3, then line 340 sends the program back to line 190 to ask for the menu selection all over again.

Lines 350 through 440 display the prompt asking for the tax filename. There are a number of ways to organize such a set of

files. If more than one person is using the computer, assign different file names, such as: D:JOHN.TAX, or D:ETHEL.TAX. Another option is to use the extender as a calendar flag, such as D:JOHN.FEB, and D:JOHN.MAR.

Line 460 is an error trap. If you try to open a file that does not exist, as one would expect to do the first time writing to a file, then that file is created. Line 460 looks for such occurrences and routes the program to lines 1060 through 1230.

Lines 490 through 530 blank the screen and print the title page, also known as the *header*.

Line 570 inputs the deduction, along with all pertinent information relating to category, and so on. Line 610 writes this information to the appropriate file.

Line 590 watches for a carriage return. After the program asks for a deduction, if you hit return, and nothing else, that signals the program you are done. Then, line 980 will close the file, and line 990 will send us back to the task menu.

Lines 650 through 760 will ask for the tax file you wish to review and then open it for a read. Line 800 sets the trap that will look for the end of the file. When it's encountered, the program lumbers off to line 920, waiting for you to hit any key, before displaying the task menu.

Line 840 checks to see if this is a cassette file about to be read. If it is, there is the little matter of some dummy data that was written during the initialization stage. If this is a cassette file, that data is dumped.

Lines 850 through 910 read twenty lines of deductions at a time. Hit any key to see the next twenty, and so on.

Lines 920 through 990 close the file and return us to the menu.

String and Numeric Variables

DEDUCTIONS$	Category of deduction.
FILENAME$	File name where information will be stored.
TAX$	Prints title at top of screen.
HEADER$	Prints items to be entered at top of screen.
PICK	Do which job in menu?
COUNT	Number of lines read and displayed from tax file.

ERROR	Kind of error found by error trap.
KEY	Any key hit yet?

The Program

```
100 REM INCOME TAX DEDUCTION RECORDER
110 DIM DEDUCTION$(35)
120 DIM FILENAME$(20)
130 DIM TAX$(17)
140 TAX$="$$  TAX BREAK  $$"
150 DIM HEADER$(39)
160 HEADER$=" DEDUCTION    ITEM    AMOUNT   DATE  "
170 REM OPEN KEYBOARD FOR READ
180 OPEN #1,4,0,"K:"
190 PRINT CHR$(125):REM BLANK THE SCREEN
200 REM DISPLAY THE TASK MENU
210 POSITION 8,2
220 PRINT TAX$
230 POSITION 2,7
240 PRINT "1. RECORD DEDUCTION(S)"
250 POSITION 2,9
260 PRINT "2. SEE DEDUCTIONS TO DATE"
270 POSITION 2,11
280 PRINT "3. QUIT SESSION"
290 POSITION 5,15
300 PRINT "(PICK 1 - 3 )";
310 REM CHOOSE ON THEN DO TASK
320 GET #1,PICK
330 ON PICK-48 GOTO 350,650,1010
340 GOTO 190
350 PRINT CHR$(125):REM BLANK THE SCREEN
360 REM UPDATE WHICH TAX FILE?
370 POSITION 5,2
380 PRINT " RECORD TAX DEDUCTION(S) "
390 POSITION 11,7
400 PRINT " EXAMPLE :  D:FEBRUARY.TAX"
410 POSITION 23,9
420 PRINT "C: (CASSETTE)"
430 POSITION 2,5
440 PRINT "WRITE WHICH TAX FILE";:INPUT FILENAME$
450 IF FILENAME$(1,1)="C" THEN 1340
460 TRAP 1060:REM BAD FILE NAME?
470 REM OPEN FOR APPEND (9)
480 OPEN #2,9,0,FILENAME$
490 PRINT CHR$(125):REM BLANK THE SCREEN
500 POSITION 12,2
510 PRINT TAX$
520 PRINT
530 PRINT HEADER$
540 PRINT
550 COUNT=COUNT+1
560 REM RECORD DEDUCTION
570 INPUT DEDUCTION$
580 REM CARRIAGE RETURN WHEN DONE
590 IF DEDUCTION$="" THEN 980
600 REM WRITE DEDUCTION TO FILE
610 PRINT #2;DEDUCTION$
620 PRINT
```

```
630 REM GET NEXT ITEM
640 GOTO 530
650 PRINT CHR$(125):REM BLANK THE SCREEN
660 REM READ WHICH TAX FILE?
670 POSITION 5,2
680 PRINT " READ EXISTING TAX RECORD(S) "
690 POSITION 10,7
700 PRINT " EXAMPLE :  D:FEBRUARY.TAX"
710 POSITION 22,9
720 PRINT "C: (CASSETTE)"
730 POSITION 2,5
740 PRINT "READ WHICH TAX FILE";:INPUT FILENAME$
750 REM OPEN FILE FOR READ (4)
760 OPEN #2,4,0,FILENAME$
770 REM ZERO THE LINE COUNT
780 COUNT=0
790 REM WATCH FOR END OF FILE
800 TRAP 920
810 PRINT CHR$(125):REM BLANK THE SCREEN
820 PRINT HEADER$
830 REM DUMP DUMMY CASSETTE DATA
840 IF FILENAME$(1,1)="C" THEN INPUT #2;DEDUCTION$
850 INPUT #2;DEDUCTION$
860 REM HOW MANY LINES PRINTED?
870 COUNT=COUNT+1
880 PRINT DEDUCTION$
890 REM 20 LINES PRINTED?
900 IF COUNT=20 THEN GET #1,KEY:COUNT=0
910 GOTO 850
920 TRAP 920
930 REM DISPLAY END OF FILE MESSAGE
940 PRINT :PRINT :PRINT
950 POSITION 12,21
960 PRINT "   END OF FILE ";
970 GET #1,KEY
980 CLOSE #2
990 GOTO 190
1000 REM QUIT SESSION ROUTINE
1010 POSITION 20,11
1020 PRINT "BYE!"
1030 POSITION 2,20
1040 END
1050 REM DOES DISK FILE EXIST?
1060 PRINT CHR$(125)
1070 ERROR=PEEK(195)
1080 IF ERROR<>170 THEN 1250
1090 POSITION 10,2
1100 PRINT FILENAME$
1110 PRINT
1120 PRINT "NOT FOUND . . ."
1130 PRINT
1140 PRINT "HIT ANY KEY TO CREATE : ";FILENAME$
1150 PRINT
1160 PRINT "HIT BREAK TO QUIT . . ."
1170 PRINT :PRINT
1180 CLOSE #2
1190 GET #1,KEY
1200 PRINT "CREATING ";FILENAME$
1210 OPEN #2,8,0,FILENAME$
1220 CLOSE #2
1230 GOTO 460
1240 REM DISPLAY ERROR MESSAGE
1250 POSITION 12,12
1260 PRINT "ERROR # ";ERROR
```

```
1270 POSITION 11,14
1280 PRINT "CORRECT ERROR"
1290 POSITION 10,16
1300 PRINT "THEN TRY AGAIN . . ."
1310 CLOSE #1
1320 END
1330 REM CASSETTE FILE UTILITY (INITIALIZE)
1340 PRINT CHR$(125):REM BLANK THE SCREEN
1350 REM PRINT INSTRUCTIONS
1360 POSITION 8,2
1370 PRINT " INITIALIZE CASSETTE FILE "
1380 POSITION 5,5
1390 PRINT "1. INSERT TAPE"
1400 POSITION 5,7
1410 PRINT "2. PRESS RECORD/PLAY"
1420 POSITION 5,9
1430 PRINT "3. HIT RETURN "
1440 REM PRINT WAIT MESSAGE
1450 POSITION 5,11
1460 PRINT "4. WAIT . . .";
1470 REM OPEN THE CASSETTE CHANNEL (WRITE)
1480 OPEN #2,8,0,"C:"
1490 REM WRITE DUMMY DATA (32 = SPACE)
1500 FOR COUNT=1 TO 127
1510 PRINT #2;CHR$(32);
1520 NEXT COUNT
1530 PRINT #2
1540 GOTO 490
```

30 / Jogger's Electronic Logbook

Are you an afficionado of "runner's high"? Perhaps you've read the earlier chapter on weight control and have chosen running as a method for achieving and maintaining physical and mental well being. Regardless of the motivation, sometimes it's nice to know where you've been. Are your times getting faster? Are you feeling stronger? How does your conditioning compare to what it was a year ago?

The jogbook will save your sport's vital statistics for later comparison. As it's set up, you can enter a day's run or view the whole logbook.

In recording the day's run, you may prefer to jot down elapsed times and type them into the computer in one sitting, every couple of days or once a week.

In the body of the program, you will be asked for how far you ran, how long it took, and how you felt afterwards. On a scale of one to ten, with ten measuring euphoria, how did you feel? Here's where there's room for modification. How did you feel before you ran? How did you feel after you ran? Which matters most to you? There's room to enter both; just separate them by a few spaces.

If you'd like to recount your exercise sessions, type in the number 2 at the menu, and the results of your miles and miles of aerobic activity will display on the screen twenty lines (days) at a time. To see each twenty days' worth of running, hit any key.

When finished either entering a run or perusing the log, the program will jump back to the menu. Typing a number 3 at the menu will end the program.

How the Program Works

Lines 110 through 180 dimension and declare the string variables. In line 120, BLANK$ consists of twenty blank spaces between those quote marks. They will be used to erase old information during the course of the program.

Line 200 opens the keyboard so we can get the user's choice of tasks.

Lines 230 through 330 format and display the menu. Line 360 sends the program to do the selected task.

Lines 380 through 450 prompt for the filename we'll be writing to. Line 460 is a safety valve, an error trap. In case this file doesn't exist yet, as would be the case the first time it's written to, line 460 will delegate that job to lines 1080 through 1110.

Line 470 opens the logbook for append. That means whatever information already exists in it will remain; anything new will be added to the end of the file.

Lines 480 through 670 ask for all of the pertinent information regarding the day's run. Line 680 writes it to the file.

When finished entering the data, hit the return key, and nothing else, after the date prompt. That will close the file and call up the task menu.

Lines 760 through 830 prompt for the jogbook to be read. Lines 840 through 950 read the information and display it on the screen, twenty lines at a time.

String and Numeric Variables

BLANK$	A string of twenty blank spaces used as an eraser.
MILES$	Number of miles run.
ET$	Elapsed time.
DATE$	Day of run.
MOOD$	Runner's feelings before, after, or both.
LINE$	Used to read back information from the file.
FILENAME$	Filename used for storing all this information.
COUNT	Controls reading and displaying twenty lines at a time from file.

CHOICE	Choice selected by the user from task menu.
KEY	Any key pressed yet?

The Program

```
100 REM THE JOGGER'S ELECTRONIC LOGBOOK
110 DIM BLANK$(20)
120 BLANK$="                    "
130 DIM MILES$(39)
140 DIM ET$(39)
150 DIM DATE$(39)
160 DIM MOOD$(39)
170 DIM LINE$(255)
180 DIM FILENAME$(20)
190 REM OPEN KEYBOARD FOR A READ
200 OPEN #1,4,0,"K:"
210 PRINT CHR$(125):REM BLANK THE SCREEN
220 REM DISPLAY THE TASK MENU
230 POSITION 12,2
240 PRINT " JOGGER LOG "
250 POKE 84,7
260 PRINT "1. ENTER DAY'S RUN"
270 POKE 84,9
280 PRINT "2. VIEW LOGBOOK"
290 POKE 84,11
300 PRINT "3. FINISHED"
310 REM GET THE CHOICE
320 POSITION 5,13
330 PRINT "PICK 1 - 3 ";:GET #1,CHOICE
340 PRINT CHR$(125):REM BLANK THE SCREEN
350 REM GO DO APPROPRIATE TASK
360 ON CHOICE-48 GOTO 380,750,1130
370 REM ENTER DAY'S STATISTICS
380 POSITION 12,2
390 PRINT " SAVE RUNNER'S DATA "
400 POSITION 5,6
410 PRINT " EXAMPLE : D:JOGGER.REC"
420 POSITION 16,7
430 PRINT "C: (CASSETTE USERS)"
440 POSITION 2,4
450 PRINT "SAVE IN FILE ";:INPUT FILENAME$
460 TRAP 1080:REM DOES FILE EXIST?
470 OPEN #2,9,0,FILENAME$
480 PRINT CHR$(125):REM BLANK THE SCREEN
490 POSITION 10,2
500 PRINT "  LOG DAY'S RUN "
510 POSITION 2,8
520 PRINT BLANK$
530 POSITION 2,5
540 PRINT "DATE";:INPUT DATE$
550 IF DATE$="" THEN 720:REM ENTER ZERO WHEN DONE
560 POSITION 2,7
570 PRINT "MILES RUN (1.3, 4.5) ";:INPUT MILES$
580 POSITION 2,9
590 PRINT "ELAPSED TIME (.5 = 30 MIN.) ";:INPUT ET$
600 POSITION 2,18
610 PRINT " NOTE :  "
620 POSITION 10,19
```

```
630 PRINT "  1 = LOW ENERGY LEVEL "
640 POSITION 10,20
650 PRINT " 10 = SUPER=CHARGED !! "
660 POSITION 2,11
670 PRINT "ENERGY LEVEL (1 TO 10) ";:INPUT MOOD$
680 PRINT #2;DATE$,MILES$,ET$,MOOD$
690 REM GET NEXT ENTRY
700 GOTO 480
710 REM DONE ENTERING, CLOSE THE FILE
720 CLOSE #2
730 REM RETURN TO TASK MENU
740 GOTO 210
750 REM VIEW THE LOGBOOK
760 POSITION 12,2
770 PRINT " READ JOGGER'S LOG "
780 POSITION 5,6
790 PRINT " EXAMPLE : D:JOGGER.REC"
800 POSITION 15,7
810 PRINT " C: For Cassette "
820 POSITION 4,4
830 PRINT "WHICH FILE ";:INPUT FILENAME$
840 PRINT CHR$(125):REM BLANK THE SCREEN
850 REM OPEN THE FILE FOR A READ
860 OPEN #2,4,0,FILENAME$
870 TRAP 970:REM WATCH FOR THE END OF FILE
880 REM READ ONE DAY'S RECORD
890 INPUT #2,MILES$
900 REM PRINT THE INFORMATION
910 PRINT MILES$
920 COUNT=COUNT+1
930 IF COUNT=20 THEN GET #1,KEY:COUNT=0
940 REM GET NEXT DAY'S RUN
950 GOTO 890
960 REM END OF FILE ROUTINE
970 CLOSE #2
980 PRINT
990 PRINT
1000 POSITION 12,22
1010 PRINT " END OF FILE ";
1020 POKE 764,255
1030 IF PEEK(764)=255 THEN 1030
1040 POKE 764,255
1050 REM RETURN TO TASK MENU
1060 GOTO 210
1070 REM DISK FILE DOES NOT EXIST, FIX!
1080 CLOSE #2
1090 OPEN #2,8,0,FILENAME$
1100 CLOSE #2
1110 GOTO 470
1120 REM FINISHED, END PROGRAM
1130 POSITION 5,12
1140 PRINT "Don't Forget To Run Tommorrow!"
1150 POSITION 2,21
1160 END
```

31 / Credit Card Manager

Do you like to keep track of purchases made with your credit card? It's easy with this diskette file manager. RUN the program and choose between RECORDing a purchase, READing the file, or QUITing. Just enter a number 1, 2, or 3. No return is necessary.

For the moment, let's assume you chose Record a Purchase, by typing the number 1. The screen will blank for a moment then ask the name under which you are storing the credit card purchase information. An example will be provided to help you. Next, a prompt will ask for the name of the item bought, how much was paid for it, and the date of the purchase.

Is there more than one item to record? If so, keep plugging in the information. When you'd like to quit, at the NAME OF ITEM prompt, hit RETURN and no other key. Control will loop back to the task menu.

This time choose the number 2, and enter the file where the credit card information is stored. Twenty lines at a time will be read from the file and displayed on the TV screen. Hit any key to scroll another twenty lines of financial facts.

When the file is empty, a message will inform you. Again, hit any key, but this time, instead of more dollars-and-cents figures, you'll see the the task menu.

If finished with all this financial wizardry, type the number 3 and the program will end.

How the Program Works

Lines 110 through 130 dimension the string variables. Line 140 opens the keyboard for a read, in order to choose one of the menu selections.

Lines 140 through 250 display and format the task menu.

Lines 250 and 260 send the program off to perform its assigned task. If the choice was anything other then the numbers 1 through 3, line 270 jumps back to line 150 and demands a menu selection all over again.

Lines 280 through 370 ask for the filename to be written to. Line 380 looks to see if it's a cassette file. If it is, it has got to be initialized. That's the function of lines 1130 through 1330.

Line 390 is an error trap. Line 400 opens the filename. If the file does not exist, as would be the case the first time you write to it, the error trap set in the previous line jumps control to lines 940 through 970. These lines create the file and patch things up with the operating system.

Lines 410 through 590 ask for the credit card information and write it to the file. Here's more detail: A prompt appears on the screen asking for the name of the item, the amount paid, and the date of purchase. See lines 470, 520, and 570. Notice how they are preceded by a POSITION command that insures these prompts always appear on the same line. Also appearing with a corresponding POSITION statement are lines 440, 500, and 550. In these lines we see the statement: PRINT CHR$(156). 156 is the ATASCII, or the computer code, the ATARI Home Computer uses to delete a line of text. That's how we erase unwanted information and keep it from ending up in a file.

Line 480, by the way, looks for a carriage return. Hit the RETURN key when you're through entering.

Line 580 writes all the information, in blocks of 128 bytes, to the file.

Lines 610 through 690 ask for the credit card file you would like to review. Line 710 opens that file. Line 720 sets the error trap to look for the end of that file.

Line 750 checks to see if we're about to read a cassette file. If so, that line will dump the dummy data we wrote to initialize it.

Lines 760 through 800 read and print the credit card file, or any other file, twenty lines at a time. Lines 820 through 920

perform one method of checking to see if any key has been pressed, before allowing another twenty lines to be displayed.

Lines 940 through 970 create a file, if you've tried to write one that doesn't exist.

Lines 990 through 1060 come at the end of file and wait for any key to be pressed before sending us back to the task menu.

Lines 1080 through 1110 are the quit routine, closing the keyboard and ending the program.

Lines 1130 through 1330 initialize a cassette file by writing dummy information to it.

String and Numeric Variables

AMOUNT$	Cost of purchase.
ITEM$	Name of item bought.
DATE$	When the purchase was made.
FILENAME$	Name of the file we will read or write to.
CHOICE	Which job we want done.
COUNT	Controls reading and printing twenty lines at a time from the file.
KEY	Any key pressed yet?

The Program

```
100 REM CREDIT CARD MANAGER
110 DIM AMOUNT$(20)
120 DIM ITEM$(255):DIM DATE$(20)
130 DIM FILENAME$(20)
140 OPEN #1,4,0,"K:"
150 PRINT CHR$(125):REM BLANK THE SCREEN
160 POSITION 7,2
170 PRINT " Credit Card Manager "
180 POKE 84,5
190 PRINT "1.  RECORD PURCHASE"
200 POKE 84,7
210 PRINT "2.  READ EXISTING FILE"
220 POKE 84,9
230 PRINT "3.  QUIT"
240 POSITION 6,13
250 PRINT "(Choose 1 - 3) ";:GET #1,CHOICE
260 ON CHOICE-48 GOTO 280,610,1080
270 GOTO 150
280 PRINT CHR$(125):REM CLEAR SCREEN
290 POSITION 12,2
```

```
300 PRINT " Record Purchase "
310 REM OPEN CASSETTE CHANNEL FOR WRITE
320 POSITION 3,7
330 PRINT " EXAMPLE : D:MASTER.CAR"
340 POSITION 14,8
350 PRINT "C: (For Cassette Users)"
360 POSITION 2,5
370 PRINT "Which File ";:INPUT FILENAME$
380 IF FILENAME$(1,1)="C" THEN 1130
390 TRAP 940
400 OPEN #2,9,0,FILENAME$
410 PRINT CHR$(125):REM BLANK THE SCREEN
420 POSITION 10,2
430 PRINT " RECORD INFORMATION "
440 POSITION 2,11:REM CHR$(156) DELETES A LINE OF TEXT
450 PRINT CHR$(156)
460 POSITION 2,11
470 PRINT "NAME OF ITEM ";:INPUT ITEM$
480 IF ITEM$="" THEN GOTO 1000
490 POSITION 2,11
500 PRINT CHR$(156)
510 POSITION 2,11
520 PRINT "AMOUNT PAID $";:INPUT AMOUNT$
530 REM RECORD PURCHASE DATE
540 POSITION 2,11
550 PRINT CHR$(156)
560 POSITION 2,11
570 PRINT "DATE OF PURCHASE ";:INPUT DATE$
580 PRINT #2;ITEM$,AMOUNT$,DATE$
590 GOTO 440
600 REM RETRIEVE RECORDS
610 PRINT CHR$(125):REM BLANK THE SCREEN
620 POSITION 13,2
630 PRINT " Read Existing File "
640 POSITION 3,7
650 PRINT " EXAMPLE : D:MASTER.CAR"
660 POSITION 14,8
670 PRINT "C: (For Cassette Users)"
680 POSITION 2,5
690 PRINT "Which File ";:INPUT FILENAME$
700 REM OPEN THE FILE FOR A READ
710 OPEN #2,4,0,FILENAME$
720 TRAP 990
730 PRINT CHR$(125):REM BLANK THE SCREEN
740 REM DUMP DUMMY LINE IF CASSETTE FILE
750 IF FILENAME$(1,1)="C" THEN INPUT #2;ITEM$
760 INPUT #2,ITEM$
770 COUNT=COUNT+1
780 PRINT ITEM$
790 IF COUNT=20 THEN GOSUB 820
800 GOTO 760
810 REM WAIT ROUTINE
820 PRINT
830 PRINT
840 POSITION 14,22
850 PRINT " Hit Any Key";
860 POKE 764,255
870 IF PEEK(764)=255 THEN 870
880 POKE 764,255
890 COUNT=0
900 PRINT
910 PRINT
920 RETURN
930 REM DISK FILE NONEXISTENT SO CREATE!
```

```
940 CLOSE #2
950 OPEN #2,8,0,FILENAME$
960 CLOSE #2
970 GOTO 400
980 REM END OF FILE ROUTINE
990 TRAP 990
1000 PRINT
1010 PRINT
1020 POSITION 14,22
1030 PRINT "  END OF FILE";
1040 GET #1,KEY
1050 CLOSE #2
1060 GOTO 150
1070 REM QUIT ROUTINE
1080 POSITION 10,20
1090 PRINT "Thank you, bye !"
1100 CLOSE #1
1110 END
1120 REM CASSETTE FILE UTILITY (INITIALIZE)
1130 PRINT CHR$(125):REM BLANK THE SCREEN
1140 REM PRINT INSTRUCTIONS
1150 POSITION 8,2
1160 PRINT " INITIALIZE CASSETTE FILE "
1170 POSITION 5,5
1180 PRINT "1. INSERT TAPE"
1190 POSITION 5,7
1200 PRINT "2. PRESS RECORD/PLAY"
1210 POSITION 5,9
1220 PRINT "3. HIT RETURN "
1230 REM PRINT WAIT MESSAGE
1240 POSITION 5,11
1250 PRINT "4. WAIT . . .";
1260 REM OPEN THE CASSETTE CHANNEL (WRITE)
1270 OPEN #2,8,0,"C:"
1280 REM WRITE DUMMY DATA (32 = SPACE)
1290 FOR COUNT=1 TO 127
1300 PRINT #2;CHR$(32);
1310 NEXT COUNT
1320 PRINT #2
1330 GOTO 410
```

FPT $14.50

Home Applications and Games for the ATARI® Home Computers

For the ATARI® 400™/800™, 600XL™, 800XL™, 1200XL™, 1400XL™, and 1450XLD™ Home Computers

Timothy P. Banse

Now, one book lets you do the two things you bought your ATARI® Home Computer for: save time and money...

• Checkbook Balancer • Budget Power • Number Averaging • Calorie Counter • Blood Alcohol Test • Crypto System • Medical History • Car Ownership Evaluation • Trip Cost Tabulator • Meal Planner • Utility Audit • Heating Loss Cost Analysis • Bulk Purchase Tabulator • Smart Typewriter – Dumb Word Processor • Carpool Worksheet • Typing Tutor • Home Inventory Log • Tax Deduction Recorder • Jogger's Electronic Logbook • Credit Card Manager

play exciting new games...

• Ghost Town Vampire Girls • Beowulf versus Grendel • Helicopter War • Jet Jockey • Bridge Buster • .44 Magnum Russian Roulette • Oracle at Delphi • I-Ching Coin Toss • Adventure Die Cast • "R" is for Red • Music Composer

and you get two easy ways to use every program:

quickly

Just type in the program listings using your computer keyboard, and run the programs "as is." It's as simple as that!

your own way

Each program is thoroughly introduced with background information, a line-by-line explanation of how the program works, and a string and numeric variable table that spells out what each variable stands for. So you get everything you need to easily modify any program to suit your own needs and tastes.

For novices and experts alike, this collection of useful and entertaining do-it-yourself programs has something for everyone!

Cover design by Steve Snider

Little, Brown and Company • Boston Toronto

ISBN 0-316-08044-6

10831400