I Speak BASIC to My Atari

Aubrey B. Jones, Jr.

For the Atari 400, 800, and 1200XL Microcomputers

A field-tested computer literacy course that introduces students to BASIC language programming.
To Alyce, Aubrey III, and Adrienne
Contents

Part 1 The Hardware (Or The "Boxes") ............................. 1
Objectives: Typical Data Processing Operation; Basic Parts of a Computer; Summary; Practice 1

Part 2 The Software ("The Program") .............................. 17
Objectives: How Humans Talk to Computers; A BASIC Program; Atari Keyboards; Atari Power-Up Rules; Summary; Practice 2

Part 3 Your First Computer Program ............................... 38
Objectives: Writing Your First Computer Program; Executing Your Program; Expanding Your Program; Listing Your Program; Ending Your Program; Summary; Practices 3, 4, 5

Part 4 More Programming Tools .................................... 58
Objectives: Mathematical Operations; Programming Mathematical Operations; A BASIC Mathematical Program—Area of a Rectangle; Print Zones; Practices 6, 7

Part 5 Scientific Notation ........................................... 79
Objectives: Scientific Notation; Review and Feedback; Practice 8

Part 6 Relational Operators and IF-THEN/GOTO Statements . 86
Objectives: Relational Operators; Using IF-THEN Statements (Conditional Branching); Using GOTO Statements (Unconditional Branching); Practices 9, 10

Part 7 Input Statements .............................................. 99
Objectives: Input Statements; Area of Rectangle Problem Revisited (Using Input Statements); String Variables; Practices 11, 12, 13

Part 8 Using the Calculator Mode and Sizing Memory .......... 112
Objectives: Bit vs. Byte; How Much Memory Is Used in BASIC Programs; Summary; Practice 14

Part 9 Using the Cassette Recorder and the Disk Drive ......... 121
Objectives: The Cassette Recorder and the Disk Drive as Input/Output Devices; Practices 15, 16, 17

Part 10 Using FOR-NEXT-STEP Statements ...................... 130
Objectives: FOR-NEXT Statements and Loops; Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops; Loop Flowcharts; Timer Loops; Practices 18, 19

Part 11 Reading Data ................................................. 146
Objectives: READ-DATA Statements; Restore Function; Practice 20
Part 12  Video Display Graphics .............................. 160
Objectives; Text and Graphic Commands: COLOR,
GRAPHICS, PLOT, POSITION, SOUND, STICK,
DRAWTO; Practice 21

Part 13  Arrays ................................................. 182
Objectives; One-Dimensional Arrays; Two-Dimensional
Arrays; DIM Statement; Summary; Practices 22, 23

Part 14  INT(X), ABS(X), and RND(X) Functions .......... 197
Objectives; INT(X) Function; ABS(X) Function; RND(X)
Function; Summary; Practices 24, 25

Part 15  Subroutines ........................................... 211
Objectives; Subroutines; ON-GOTO; ON-GOSUB;
Summary; Practices 26, 27

Extra Practices ............................................... 227
PART 1

The Hardware (Or The "Boxes")

What You Will Learn

1. That the computer is a valuable tool that can solve problems, print words, draw pictures, store information, retrieve information, compare information, play games, and do many other things to help you in everyday life.

2. That people control computers and that computers cannot think (despite what you might have heard).

3. To identify and explain the basic parts of a computer and relate them to a "box diagram" of a general purpose computer.

4. To identify and explain the function of the basic parts of an Atari microcomputer.

5. To define and explain the terms hardware, software, microcomputer, microprocessor, RAM, ROM, processor, input unit, output unit, memory, and binary.

6. That computers are simple and easy to use; and above all that computers are fun!
Welcome to the World of Computers!

People Control Computers!

Computers Can’t Think!
Typical Data Processing Operation

"Box" Diagram

Examples of Data Processing Operation

**INPUT**

1. NUMBER HR WORKED (40 hr)
2. RATE/HR ($3.00/HR)

**PROCESSING (CALCULATE)**

40hr X 3.00/hr = $120

**OUTPUT**

PAY TO XXXX
$120

**INPUT**

1. TOTAL NUMBER OF STUDENTS IN CLASS
2. GRADES FOR EACH STUDENT

**PROCESSING (CALCULATE)**

1. TOTAL ALL GRADES
2. TOTAL ALL STUDENTS
3. TOTAL GRADES
   TOTAL STUDENTS

**OUTPUT**

AVG GRADE
BOX Diagram
Showing
Basic Parts of a Computer

- STORAGE UNIT

- INPUT UNIT
- PROCESSOR UNIT
- OUTPUT UNIT
Stores or Remembers

- Storage unit (memory)
  - Stores both information and instructions until needed (requested)
Interprets, Controls, & Calculates

- PROCESSOR UNIT
  - INTERPRETS (DECODES) INSTRUCTIONS AND REGULATES (CONTROLS) THEIR EXECUTION
  - PERFORMS ALL OF THE CALCULATIONS
Box Diagram of a Basic Computer System

- **INPUT UNIT**
  - DATA
  - INSTRUCTIONS

- **PROCESSOR UNIT**
  - INTERPRETS & CONTROLS
  - PERFORMS CALCULATIONS

- **OUTPUT UNIT**
  - RESULTS
  - ANSWERS

- **STORAGE UNIT**
  - STORES OR REMEMBERS
What We Have Learned

- **INPUT** → PROVIDES INSTRUCTIONS AND DATA
- **STORAGE** → STORES OR REMEMBERS (MEMORY)
- **PROCESSOR** → INTERPRETS, CONTROLS, & CALCULATES
- **OUTPUT** → PROVIDES ANSWERS AND RESULTS
"Human Computer"
Man Can Think But Computer Can't!

EXECUTE (PERFORM WHATEVER OPERATION IS NECESSARY)

CONTROLS
DECODES INSTRUCTIONS

INTERPRETS (INPUTS)

MEMORY (REMEMBERS INPUTS)

INPUT

INPUT (SOUNDS)

INPUT (SIGHTS)

INPUT (SMELLS)

INPUT/OUTPUT
Some Terms You Should Know

- MICROPROCESSOR
- MICROCOMPUTER
- RAM
- ROM

- MICRO = Very small
- MICROPROCESSOR = Very small processor
- RAM = Random access memory
  - CAN BE changed by the user
  - Information stored in RAM will be destroyed if power fails or turned-off (volatile)
- ROM = Read only memory
  - CANNOT be changed by the user
  - Information stored in ROM is not destroyed if power fails or is turned-off (non-volatile)
  - Control program (BASIC interpreter) stored here
Box Diagram of a Microcomputer

- STORAGE UNIT

- INPUT UNIT

• MICROPROCESSOR UNIT

• OUTPUT UNIT
Basic Components of the Atari Computer

- KEYBOARD
- INPUT
- MEMORY
- MICROPROCESSOR
- TAPE RECORDER (I/O DEVICE)
- PRINTER OUTPUT
- VIDEO DISPLAY OUTPUT
Atari 400 Microcomputer

Atari 1200XL Microcomputer

Courtesy of Atari, Inc.
Atari 800 Microcomputer

Atari 800 with Cartridge Slots Visible

Courtesy of Atari, Inc.
What We Have Learned

<table>
<thead>
<tr>
<th>DATA PROCESSING OPERATIONS</th>
<th>BASIC COMPUTER PARTS</th>
<th>MICROCOMPUTER PARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>INPUT UNIT</td>
<td>INPUT UNIT</td>
</tr>
<tr>
<td>PROCESSING</td>
<td>PROCESSOR UNIT + MEMORY UNIT</td>
<td>MICROPROCESSOR + MEMORY</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>OUTPUT UNIT</td>
<td>OUTPUT UNIT</td>
</tr>
</tbody>
</table>
PRACTICE 1

Box Diagram of a Computer

1. Draw the BOX DIAGRAM of a BASIC computer.
   a. Label each box with the correct name.
   b. List the functions of each box.
PART 2

The Software (The "Program")

What You Will Learn

1. To define the terms hardware, software, BASIC, binary, and interpreter, and to relate them to computers.
2. That computers speak a foreign language: machine language.
3. How humans talk to computers via a programming language called BASIC.
4. To identify the principal parts of a BASIC program.
5. To identify and explain the purpose of all the keys on the Atari keyboard.
6. How to connect and power up an Atari microcomputer.
Box Diagram of a Basic Computer System

- INPUT UNIT
  - DATA
  - INSTRUCTIONS

- STORAGE UNIT
  - STORES OR REMEMBERS

- PROCESSOR UNIT
  - INTERPRETS & CONTROLS
  - PERFORMS CALCULATIONS

- OUTPUT UNIT
  - RESULTS
  - ANSWERS
More Terms You Should Know

• HARDWARE
  — THE COMPUTER AND COMPUTER RELATED EQUIPMENT (THE BOXES)

• SOFTWARE
  — THE INSTRUCTIONS FOR THE COMPUTER (THE PROGRAM)
Computers Speak a Foreign Language!
(No Speak English, French, German, Spanish, or Any Other Natural Language)
• COMPUTERS SPEAK IN *MACHINE LANGUAGE*

  — MACHINE LANGUAGE IS A FORM OF *BINARY CODING*

  — BINARY IS A WORD DENOTING “TWO”

  — MACHINE LANGUAGE USES TWO BASIC SYMBOLS: “0” AND “1”
How Humans Talk to Computers

- **NATURAL LANGUAGE**
  - ENGLISH — THINK

- **PROGRAMMING LANGUAGE**
  - BASIC — WRITE

- **TRANSLATOR**
  - INTERPRETER — INTERPRET

- **MACHINE LANGUAGE CODE**
  - MACHINE — EXECUTE
- **BASIC**
  (Beginner's all-purpose symbolic instruction code)
  - Popular programming language for writing instructions to the computer

- **INTERPRETER**
  - Translates BASIC into machine code
  - (You really don't have to know anything about an interpreter since it is used automatically when you run a BASIC program)
  - Located in the ROM in Atari
To Program You Must Learn the Language First!

A Comparison between English and BASIC

**ENGLISH LANGUAGE**
- Words  
  - Used to make sentences
- Sentences  
  - Used to make paragraphs
- Paragraphs  
  - Lengths vary
- Commands  
  - Can be one word  
    - e.g., STOP! HALT!
- Sentence Numbers  
  - Optional (seldom used)

**BASIC PROGRAMMING LANGUAGE**
- Key Words  
  - Used to make statements
- Statements  
  - Used to make programs
- Programs  
  - Lengths vary
- Commands  
  - Executed immediately  
    - e.g., NEW, LIST, RUN
- Line Numbers  
  - Must be used for each statement
Learning a New Vocabulary

Here Are the Key Words and Commands You’ll Learn:

<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>NEW</td>
</tr>
<tr>
<td>END</td>
<td>LIST</td>
</tr>
<tr>
<td>LET</td>
<td>RUN</td>
</tr>
<tr>
<td>INPUT</td>
<td>CONT</td>
</tr>
<tr>
<td>GOTO</td>
<td></td>
</tr>
<tr>
<td>IF ... THEN</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td>FOR ... NEXT</td>
<td></td>
</tr>
<tr>
<td>READ-DATA</td>
<td></td>
</tr>
</tbody>
</table>
Commands vs. Statements

COMMANDS

— Executed as soon as you type them and press RETURN

STATEMENTS

— Put into programs and are only executed after you type the command RUN and press RETURN
A BASIC Program

<table>
<thead>
<tr>
<th>LINE NUMBER</th>
<th>KEY WORD</th>
<th>OTHER PART OF THE STATEMENT</th>
<th>&quot;LOOK AT&quot; REQUEST*</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>PRINT</td>
<td>&quot;HELLO THERE&quot;</td>
<td>RETURN</td>
</tr>
<tr>
<td>20</td>
<td>PRINT</td>
<td>&quot;YOUR NAME&quot;</td>
<td>RETURN</td>
</tr>
<tr>
<td>30</td>
<td>END</td>
<td></td>
<td>RETURN</td>
</tr>
</tbody>
</table>

**COMMAND**

RUN

*Pressing the [RETURN] key tells the computer to "LOOK AT" (and store) what you have just typed. You must press this key after each statement or command.
Line Numbers

- Serve as a guide to the computer in running the program.
- Tell the computer in what order it should carry out your instructions.
- Computer will start executing at lowest numbered line unless told to start elsewhere.
- Normally are multiples of 5's, 10's, or some other multiples to leave space for inserting new program lines between old one.
- Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4, etc.), don’t do it!
Key Words

— Never used alone
— Need line number
— Always part of a BASIC statement that has some other part to it*
— Executed only after command RUN is typed and \texttt{RETURN} key is pressed

*To the purist, we know that key words like END and STOP can be used alone; but you still need line numbers, and you must type \texttt{RUN} and press \texttt{RETURN} to execute.
What We Have Learned

• Key words
  — Used to make statements

• Statements
  — Must have line numbers and key words
  — Used to make programs

• Programs
  — May vary in length

• Commands
  — Executed as soon as you type them and press RETURN
KEYBOARD FOR ATARI 400 AND ATARI 800

KEYBOARD FOR ATARI 1200 XL
### Special Function Keys on the Atari Keyboard

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CTRL</strong></td>
<td>• Stands for CONTROL. Several keys have an additional function that is obtained by holding down the <strong>CTRL</strong> key while the other keys are pressed.</td>
</tr>
<tr>
<td></td>
<td>— Control characters are special characters that appear on the screen and are used to make special displays. They are also used to edit programs.</td>
</tr>
<tr>
<td><strong>ESC</strong></td>
<td>• Stands for ESCAPE. It is used to perform special functions in a program.</td>
</tr>
<tr>
<td></td>
<td>— <strong>ESC</strong> unlike <strong>CTRL</strong> key does not have to be held down while typing another key. (Forget about EDIT mode for now.)</td>
</tr>
<tr>
<td><strong>SYSTEM RESET</strong></td>
<td>• Press this key if your Atari computer does not respond correctly to your instructions. (If this does not work try turning your Atari on and off again. Of course, if you do this you will lose your program.) Caution should be exercised, especially if a disk drive is active, since it could possibly lose or damage data.</td>
</tr>
<tr>
<td><strong>INSERT</strong></td>
<td>• Pressing this key while pressing <strong>SHIFT</strong> inserts a line.</td>
</tr>
<tr>
<td></td>
<td>• Pressing this key while pressing <strong>CTRL</strong> inserts a space.</td>
</tr>
<tr>
<td><strong>400/800</strong></td>
<td>• Cause display to show characters in inverse.</td>
</tr>
<tr>
<td><strong>1200</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CAPS LOWR</strong></td>
<td>• Causes the Atari to display text in upper or lower case.</td>
</tr>
<tr>
<td><strong>RETURN</strong></td>
<td>• Causes the computer to &quot;look at&quot; line you just typed in and to act accordingly. This key must be pressed each time you want to enter a line from the keyboard.</td>
</tr>
<tr>
<td></td>
<td>— <strong>RETURN</strong> also causes the cursor to &quot;RETURN&quot; to the screen's left edge (a line down from where it was).</td>
</tr>
</tbody>
</table>
### Special Function Keys on the Atari Keyboard (cont’d)

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SHIFT</strong></td>
<td>• Some keys have two characters printed on them. This key permits you to type upper characters such as quotes (&quot;). Hold down <strong>SHIFT</strong> key while typing key with two symbols if you want to type the upper symbol.</td>
</tr>
<tr>
<td><strong>CLEAR &lt;</strong></td>
<td>• Holding down <strong>SHIFT</strong> and pressing <strong>CLEAR &lt;</strong> will clear the screen.</td>
</tr>
<tr>
<td><strong>DELETE BACK S</strong></td>
<td>• Deletes the character at the location of the cursor and moves the cursor one space left.</td>
</tr>
<tr>
<td></td>
<td>• Holding down <strong>SHIFT</strong> and pressing <strong>DELETE BACK S</strong> will delete the entire line, no matter where the cursor is on the line. Cursor then returns to first position on line.</td>
</tr>
<tr>
<td><strong>BREAK</strong></td>
<td>• Stops program execution.</td>
</tr>
<tr>
<td><strong>CLR SET TAB</strong></td>
<td>• Pressing this key while pressing <strong>SHIFT</strong> sets the TAB position.</td>
</tr>
<tr>
<td></td>
<td>• Pressing this key while pressing <strong>CTRL</strong> clears the TAB position.</td>
</tr>
<tr>
<td></td>
<td>• Pressing this key alone moves the cursor to the next TAB setting.</td>
</tr>
</tbody>
</table>
Control (CTRL) Key Functions

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL</td>
<td>• Stands for “CONTROL.” Holding down this key while other keys are pressed causes the computer to perform different actions. Here are some examples:</td>
</tr>
<tr>
<td>CTRL 1</td>
<td>— Stops a program listing. To restart the program listing, type CTRL 1 again.</td>
</tr>
<tr>
<td>CTRL 2</td>
<td>— Causes the computer to “BEEP.” CTRL 2 is called a “BELL” because the present keyboard design is based on the teletype where CONTROL 2 actually rings a bell.</td>
</tr>
<tr>
<td>DELETE BACK S</td>
<td>— Deletes next character to the right of cursor but does not affect cursor position.</td>
</tr>
</tbody>
</table>

Cursor Positioning

Cursor positioning allows the cursor to be moved anywhere on the screen without affecting text. This is very useful for editing without having to retype the entire program line.

To use the arrow keys, CTRL must be pressed while pressing the particular arrow key.

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL</td>
<td>— Moves the cursor left.</td>
</tr>
<tr>
<td>CTRL</td>
<td>— Moves the cursor up.</td>
</tr>
<tr>
<td>CTRL</td>
<td>— Moves the cursor down.</td>
</tr>
<tr>
<td>CTRL</td>
<td>— Moves the cursor right.</td>
</tr>
</tbody>
</table>
Atari Power-Up Rules

ACTION

1. Make certain system is connected properly. (This procedure assumes you are not using a disk.)

2. If the tape recorder is connected, it should be in the STOP mode.

3. Turn on the video display and set the RF modulator to "Computer." (Make certain that channel selection on the television knob matches that on the computer.)

4. Turn on the Atari. The switch is on the right side of the computer next to where the power cord plugs in. Push this switch into the upward position.

5. The display should appear as shown. — (The inverse square is called the "cursor.")

6. If your display does not look as shown in the description above, do the following:
   (a). If your Atari doesn't seem to respond correctly, a press of the SYSTEM RESET key will usually remedy the problem.
   (b). If pressing the SYSTEM RESET doesn't work, turning the Atari off and then turning it back on again will probably correct the problem.
Getting It Together

• STEP 1 — WRITE YOUR PROGRAM
• STEP 2 — GET THE COMPUTER READY
• STEP 3 — ENTER YOUR BASIC PROGRAM
• STEP 4 — RUN YOUR PROGRAM
• STEP 5 — SIGN OFF
PRACTICE 2

Becoming Familiar with the Atari

Become familiar with the Atari by doing the following (you should actually go through every step):

1. Power up (turn on) the Atari using the power-up rules (see page 35).
2. How many power buttons did you have to press? ___________________
3. Where were the buttons located? ___________________
4. Where is the [SYSTEM RESET] button located? ___________________
5. Where is the power indicator located? ___________________
   a. How many [SHIFT] keys are there on the keyboard? ___________________
   b. Hold down the [SHIFT] key and press every key that has a second character on the key (e.g. ! and #). What happened? _______________
   c. What happened when you held down the [SHIFT] key and pressed [2]? ___________________
7. Locate the [DELETE BACK S] key.
   a. Press the [SHIFT CLEAR] keys. (This should clear the screen with the exception of the cursor (■).)
   b. Type the following (just as shown)
      
      PRINT "CAT" ■
      
      cursor
      
      What does the cursor do each time you type a character? _______________
   c. Press the [RETURN] key. What happened? ___________________
   d. Try additional examples until you feel comfortable. Use the [DELETE BACK S] key.
8. Locate the [CTRL] and [ESC] keys. You will learn more about them later.
PART 3

Your First Computer Program

What You Will Learn

1. To enter and run your first BASIC program.

2. To explain the purpose and use of the following BASIC commands: LIST, NEW, RUN.

3. To explain the purpose and use of the following key words: PRINT, PRINT (for spacing), REM, END.

4. To explain the purpose and use of the following special function keys:
   
   `CTRL`, `RETURN`, `SHIFT`, `SYSTEM RESET`, `ESC`, `DELETE`, `BACK S`

5. To explain the purpose and use of the following miscellaneous points:
   • cursor, " " (quotes), line numbers, system-reset button, power-up rules.
## Special Function Keys on the Atari Keyboard

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
</table>
| **CTRL** | • Stands for CONTROL. Several keys have an additional function that is obtained by holding down the **CTRL** key while the other keys are pressed.  
  — Control characters are special characters that appear on the screen and are used to make special displays. They are also used to edit programs. |
| **ESC**  | • Stands for ESCAPE. It is used to perform special functions in a program.  
  — **ESC** unlike **CTRL** key does not have to be held down while typing another key. (Forget about EDIT mode for now.) |
| **SYSTEMRESET** | • Press this key if your Atari computer does not respond correctly to your instructions. (If this does not work try turning your Atari on and off again. Of course, if you do this you will lose your program.)  
  Caution should be exercised, especially if a disk drive is active, since it could possibly lose or damage data. |
| **INSERT** | • Pressing this key while pressing **SHIFT** inserts a line.  
  • Pressing this key while pressing **CTRL** inserts a space. |
| **RETURN** | • Causes display to show characters in inverse. |
| **CAPSLOWR** | • Causes the Atari to display text in upper or lower case. |
| **RETURN** | • Causes the computer to “look at” line you just typed in and to act accordingly. This key must be pressed each time you want to enter a line from the keyboard.  
  — **RETURN** also causes the cursor to “RETURN” to the screen’s left edge (a line down from where it was). |
### Special Function Keys on the Atari Keyboard (cont’d)

<table>
<thead>
<tr>
<th>KEY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SHIFT</strong></td>
<td>• Some keys have two characters printed on them. This key permits you to type upper characters such as quotes (&quot;). Hold down <strong>SHIFT</strong> key while typing key with two symbols if you want to type the upper symbol.</td>
</tr>
<tr>
<td><strong>CLEAR</strong></td>
<td>• Holding down <strong>SHIFT</strong> and pressing <strong>CLEAR</strong> will clear the screen.</td>
</tr>
<tr>
<td><strong>DELETE BACK S</strong></td>
<td>• Deletes the character at the location of the cursor and moves the cursor one space left.</td>
</tr>
<tr>
<td></td>
<td>• Holding down <strong>SHIFT</strong> and pressing <strong>DELETE BACK S</strong> will delete the entire line, no matter where the cursor is on the line. Cursor then returns to first position on line.</td>
</tr>
<tr>
<td><strong>BREAK</strong></td>
<td>• Stops program execution.</td>
</tr>
<tr>
<td><strong>CLR SET TAB</strong></td>
<td>• Pressing this key while pressing <strong>SHIFT</strong> sets the TAB position.</td>
</tr>
<tr>
<td></td>
<td>• Pressing this key while pressing <strong>CTRL</strong> clears the TAB position.</td>
</tr>
<tr>
<td></td>
<td>• Pressing this key alone moves the cursor to the next TAB setting.</td>
</tr>
</tbody>
</table>
(Review)
Control (CTRL) Key Functions

KEY  FUNCTION
CTRL  • Stands for “CONTROL.” Holding down this key while other keys are pressed causes the computer to perform different actions. Here are some examples:

CTRL 1 — Stops a program listing. To restart the program listing, type CTRL 1 again.

CTRL 2 — Causes the computer to “BEEP.” CTRL 2 is called a “BELL” because the present keyboard design is based on the teletype where CONTROL 2 actually rings a bell.

CTRL DELETE BACK S — Deletes next character to the right of cursor but does not affect cursor position.

Cursor Positioning

Cursor positioning allows the cursor to be moved anywhere on the screen without affecting text. This is very useful for editing without having to retype the entire program line.

To use the arrow keys, CTRL must be pressed while pressing the particular arrow key.

KEY  FUNCTION
CTRL — Moves the cursor left.
CTRL — Moves the cursor up.
CTRL — Moves the cursor down.
CTRL — Moves the cursor right.
(Review)

Atari Power-Up Rules

**ACTION**

1. Make certain system is connected properly. (This procedure assumes you are not using a disk.)

2. If the tape recorder is connected, it should be in the **STOP** mode.

3. Turn on the video display and set the RF modulator to "Computer." (Make certain that channel selection on the television knob matches that on the computer.)

4. Turn on the Atari. The switch is on the right side of the computer next to where the power cord plugs in. Push this switch into the upward position.

5. The display should appear as shown. →
   (The inverse square is called the "cursor.")

6. If your display does not look as shown in the description above, do the following:
   
   (a). If your Atari doesn’t seem to respond correctly, a press of the **SYSTEM RESET** key will usually remedy the problem.

   (b). If pressing the **SYSTEM RESET** doesn’t work, turning the Atari off and then turning it back on again will probably correct the problem.
Typical Display Readout

10 PRINT "HELLO THERE"

20 PRINT "YOUR NAME"

30 END

RUN
Writing Your First Computer Program

YOUR ACTION

1. Before you start typing your program, always type NEW and press the RETURN key.

2. Type the line exactly as shown:  

3. Use SHIFT key to type the upper characters like the quotation marks (" ) and the exclamation point (!).

4. Do not press RETURN key yet!

5. Go back and examine your typed line very carefully. Did you make a mistake? If you did, just press while holding down the CTRL key.

6. Is everything OK? If it is, you can press RETURN. (This tells the computer to "look at" what you just typed in).

7. When the cursor appears as shown, the computer is saying, "It's your turn . . . I'm waiting for you."

NOTE

A Insert student's name

Go to next page
Common Errors

- Missing quotes (")
- Too many quotes
- Forgot the key word PRINT
- Forgot the line number
- Forgot to press RETURN
- Used the character "O" for the number "ZERO" (Ø).

(Note: A slash is used to help you to recognize a zero.)
Writing Your First Computer Program — Almost? (Errors)

PROBLEM
(You Forgot to Follow Instructions)

1. MISSING QUOTES (") — You forgot to enclose everything after the word PRINT in quotation marks. (If you want something printed, don’t forget the quotation marks!)

2. TOO MANY QUOTATION MARKS — You typed too many. (That won’t work either!)

3. FORGOT THE KEY WORD PRINT — You forgot to type PRINT. (How will the computer know you want to print if you don’t tell it to print?)

4. FORGOT TO TYPE THE LINE NUMBER (10) — Line numbers tell the computer where to start. The computer always starts executing from the lowest numbered line unless you tell it to start elsewhere. (We will show you how to tell the computer to start at another line later. Keep the faith!)

SOLUTION

• If you have already pressed [RETURN], you must retype the entire line to correct your error. Here is how you do it:

  • Type in the same line number you wish to change (10 in this case). That is, if you want the computer to replace that line with the corrected line.

  • Next, retype the line exactly as shown on previous page. (But follow directions this time, Dummy!)

  • Then, check line over for errors.

  • If everything is OK, don’t forget to press [RETURN]! When you press [RETURN] it tells the computer to “look at” what you just typed and to act accordingly.

Read this page if you had any errors! Then correct your errors before going to the next page!
### Executing Your Program

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Tell the computer to execute or run your program. The command for this is simple: RUN.</td>
<td>HELLO THERE NAME!</td>
</tr>
<tr>
<td>2. So type <strong>RUN</strong> and press [RETURN].</td>
<td>READY</td>
</tr>
<tr>
<td>3. If you made no mistakes, the display will read:</td>
<td></td>
</tr>
<tr>
<td>4. If it did not work, try again (i.e., check your program for errors).</td>
<td></td>
</tr>
<tr>
<td>5. If it did work, let out a yell, “HEY, I CAN DO IT TOO!”</td>
<td></td>
</tr>
</tbody>
</table>

Go to next page (if you completed this one OK)
**Using the Cursor Keys to Save Time**

**YOUR ACTION**

1. You typed Line 10 as shown but have *not* pressed [RETURN] (cursor at the end of that line indicates you have not pressed [RETURN]).

2. You wish to change the “B” to a “D” or to PRINT AUDREY. So you press the [CTRL] + keys to move the cursor to the left one space at a time.

3. Now type “D” but *don’t* press [RETURN] yet. (Note that the cursor has moved to the next letter “R.”)

4. If you have finished typing the line and everything is correct, press [RETURN]. (Note that after you press [RETURN] the cursor moved to the beginning of the next line.)

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. You typed Line 10 as shown but have <em>not</em> pressed [RETURN] (cursor at the</td>
<td>10 PRINT “AUBREY” [RETURN]</td>
</tr>
<tr>
<td>end of that line indicates you have not pressed [RETURN]).</td>
<td>(cursor)</td>
</tr>
<tr>
<td>2. You wish to change the “B” to a “D” or to PRINT AUDREY. So you press the</td>
<td>10 PRINT “AUDREY”</td>
</tr>
<tr>
<td>[CTRL] + keys to move the cursor to the left one space at a time.</td>
<td>(cursor)</td>
</tr>
<tr>
<td>3. Now type “D” but <em>don’t</em> press [RETURN] yet. (Note that the cursor has</td>
<td>10 PRINT “AUDREY” [RETURN]</td>
</tr>
<tr>
<td>moved to the next letter “R.”)</td>
<td>(cursor)</td>
</tr>
<tr>
<td>4. If you have finished typing the line and everything is correct, press</td>
<td>10 PRINT “AUDREY”</td>
</tr>
<tr>
<td>[RETURN]. (Note that after you press [RETURN] the cursor moved to the</td>
<td>READY</td>
</tr>
<tr>
<td>beginning of the next line.)</td>
<td>![Cursor at the end of the next line]</td>
</tr>
</tbody>
</table>
# Some Helpful Keys and Commands to Remember

<table>
<thead>
<tr>
<th>ACTION</th>
<th>KEY(S) TO PRESS</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Enter data</td>
<td>RETURN</td>
<td>—</td>
</tr>
<tr>
<td>• Clear the screen</td>
<td>Hold down <strong>SHIFT</strong> and press <strong>CLEAR</strong></td>
<td>PRINT CHR$(125)</td>
</tr>
<tr>
<td>• Stop the program execution</td>
<td><strong>BREAK</strong></td>
<td>STOP</td>
</tr>
<tr>
<td>• Continue program execution</td>
<td>Type <strong>CONT</strong>, then press <strong>RETURN</strong></td>
<td>CONT</td>
</tr>
<tr>
<td>• Stop program listing</td>
<td>Press <strong>CTRL</strong> and <strong>1</strong></td>
<td>—</td>
</tr>
<tr>
<td>• Continue program listing</td>
<td>Press <strong>CTRL</strong> and <strong>1</strong></td>
<td>—</td>
</tr>
<tr>
<td>• Backspace</td>
<td><strong>DELETE BACK S</strong></td>
<td>—</td>
</tr>
<tr>
<td>• Type upper symbol on key</td>
<td>Press <strong>SHIFT</strong> and, <strong>desired key</strong></td>
<td>—</td>
</tr>
<tr>
<td>• Reset</td>
<td><strong>SYSTEM RESET</strong></td>
<td>—</td>
</tr>
</tbody>
</table>
Expanding Your Program

YOUR ACTION

1. You now have a program in the computer. (Unless you turned it off. If you did, retype line as shown):

2. Type in line 20 exactly as shown:

3. Check your new line (20) very carefully, especially the quotation marks.

4. Everything OK? Press RETURN. (Remember, always press RETURN if you want the computer to look at what you typed.)

5. Let's run your program. Type RUN and press RETURN.

6. If you did it right, the screen will read:

7. If it did not work, check your program for errors.

Go to next page
Using the Print Statement for Spacing

YOUR ACTION

1. Look at your video display. Would you like more space between the two lines? OK, this is how you do it.

2. Type in a new line as shown and then press RETURN.

3. Now type RUN and press RETURN.

4. WOW! A PRINT "nothing" puts a space between what you told the computer to print in Lines 10 and 20.

5. Observe that the PRINT statement (Line 15) was placed between Lines 10 and 20. Since you were smart enough to number your lines by 10's, it was much easier to modify your program. (That's because you left room to insert new lines between the old ones.) Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4), don't do it.

Go to next page
Inserting Remarks into a Program  
(But Not Printing Them Out)

**YOUR ACTION**

1. Another important key word is REM, which stands for remark. It is often convenient to insert remarks into a program. The main reason for inserting remarks is so you or someone else can refer to them later and know what the program is for and how it is used.

2. When you tell the computer to execute the program by typing **RUN** and pressing **RETURN**, it will skip right over any number line that begins with the key word REM. The REM statement will have no effect on the program. (Let's see about that!)

3. Type Line 5 exactly as shown and then press **RETURN**

4. Type **RUN** and press **RETURN**.

5. It is the same as before (REM statement was not printed).

Go to next page

<table>
<thead>
<tr>
<th><strong>DISPLAY</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>5 REM MY FIRST COMPUTER PROGRAM</td>
</tr>
<tr>
<td>HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR!</td>
</tr>
</tbody>
</table>
Listing Your Program
(Looking At Your Program to See What It Contains)

YOUR ACTION*

1. To list your program is easy.
The command is LIST.
2. Now you type LIST and press RETURN:

3. Also, you might only want to list one line. Type LIST 20 and press RETURN and the screen will display:

4. You might also want to list several program lines, starting at one line and ending at another. For example, type LIST 10, 20 and RETURN.

*Type SHIFT CLEAR and press RETURN so you can start with a clean display.

Go to next page
Ending Your Program

YOUR ACTION

1. The end of a program is the last statement you want the computer to execute. Most computers require you to place an END statement after this point, so the computer will know it is finished. However, the Atari does not require an END statement. (Other computers might require it though.)

2. Let's add an END statement to your program. Type and enter: ~

3. Now type RUN and press [RETURN].

4. No change from before! The program ended, but it did not print "END."

5. Let's make it print the END. (How do we do that?)

6. Oh, I remember! We need a PRINT statement. So let's try it. Type and enter: ~

7. Now RUN your program.

8. IT WORKED AGAIN! (If not, check the program.)

9. Note that there is no space between THE END and the line above it. Why? Because you did not tell the computer to put a space between them!
Learned in This Session

<table>
<thead>
<tr>
<th>COMMANDS*</th>
<th>KEY WORDS**</th>
<th>MISCELLANEOUS</th>
<th>SPECIAL FUNCTION KEYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• SHIFT CLEAR &lt;</td>
<td>PRINT &quot;MESSAGE&quot;</td>
<td>• CURSOR</td>
<td>CTRL CONTROL</td>
</tr>
<tr>
<td>• LIST</td>
<td>PRINT (SPACE)</td>
<td>&quot; &quot; QUOTATION MARKS</td>
<td>ESC ESCAPE</td>
</tr>
<tr>
<td>— LIST MM</td>
<td>REM</td>
<td>LINE NUMBERING</td>
<td>RETURN</td>
</tr>
<tr>
<td>• NEW</td>
<td>END</td>
<td>• KEYBOARD LAYOUT</td>
<td>SHIFT</td>
</tr>
<tr>
<td>• RUN</td>
<td></td>
<td>• ATARI POWER-UP RULES</td>
<td>DELETE BACK S</td>
</tr>
<tr>
<td>— RUN MM</td>
<td></td>
<td></td>
<td>SYSTEM RESET</td>
</tr>
</tbody>
</table>

* Executed as soon as you type them and press RETURN
** Used to make statements. Statements are executed after you type RUN and press RETURN

NOTE: If you don’t understand everything on this page, stop!
Go back over this session until you understand it thoroughly!
MM = Any line number (e.g., 10, 20, 30, etc.)
Assignment* 3-1

1. WRITE* A PROGRAM TO PRINT ON SEPARATE LINES
   A. Your Name
   B. Your Entire Address
   C. Your Telephone Number

2. EXPAND* YOUR PROGRAM TO INCLUDE THE FOLLOWING:
   A. Remark Statement to Describe Your Program
   B. Spacing between Each of the Lines Displayed (Printed)
   C. Include an End Statement

3. TYPE YOUR PROGRAM AND PRESS [RETURN]

4. RUN YOUR PROGRAM

5. LIST YOUR PROGRAM

* WRITE YOUR PROGRAM ON PAPER AND GET IT CHECKED BY YOUR TEACHER FIRST.
PRACTICE 3

Writing and Running Your First Program

1. Write a program to PRINT the following:
   a. Your name (first and last)
   b. Your school's name
   c. Your teacher's name
2. Enter and RUN it.

PRACTICE 4

Inserting Remarks and Spacing into Your Program

1. If you have erased the program from Practice 3, rewrite the program and do the following:
   (If you still have the program from Practice 3 in the computer, you do not have to rewrite the program.)
   a. Add a new program line with a remark statement to your program (any remarks you want to make).
   b. Have the computer insert one space between your name and your school's name in the output on the display (that is, you add the necessary program line).
   c. Have the computer insert two spaces between your school's name and your teacher's name in the output on the display.

PRACTICE 5

Listing and Ending Your Program

1. Rewrite the program from Practice 4 and do the following (Again, if you have the program in the computer, you don't have to rewrite it. But in case you don't know what is in the computer, just type NEW and rewrite the program.):
   a. Add an END statement to tell the computer it is the end of your program.
   b. Add a statement to have your computer PRINT "The END."
   c. RUN your program.
2. List your program.
   a. How large is your program now? (How many lines?)
   b. Copy the program in your notebook.
PART 4

More Programming Tools

What You Will Learn

1. To enter and run more BASIC programs: mathematical programs, area of rectangle program.

2. To explain the order of mathematical operations using the M.D.A.S. rule.

3. To explain the purpose and use of the keyword: LET.

4. To explain the purpose and use of the BASIC mathematic operators: multiply (*), divide (/), add (+), subtract (−), exponentiate or raising a number to a power (^).

5. To explain the function and use of commas, semicolons, and print zones.

6. To list and identify variables that can be used with Atari BASIC.
Review of Part 3

<table>
<thead>
<tr>
<th>COMMANDS*</th>
<th>KEY WORDS**</th>
<th>MISCELLANEOUS</th>
<th>SPECIAL FUNCTION KEYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• SHIFT</td>
<td>PRINT &quot;MESSAGE&quot;</td>
<td>□ CURSOR</td>
<td>CTRL CONTROL</td>
</tr>
<tr>
<td>CLEAR</td>
<td>PRINT (SPACE)</td>
<td>&quot; &quot; QUOTATION MARKS</td>
<td>ESC ESCAPE</td>
</tr>
<tr>
<td>• LIST</td>
<td>REM</td>
<td>LINE NUMBERING</td>
<td>RETURN</td>
</tr>
<tr>
<td></td>
<td>END</td>
<td></td>
<td>SHIFT</td>
</tr>
<tr>
<td>• NEW</td>
<td>** Used to make statements. Statements are executed after you type RUN and press RETURN</td>
<td>● KEYBOARD LAYOUT</td>
<td>DELETE BACK S</td>
</tr>
<tr>
<td>• RUN</td>
<td></td>
<td>● ATARI POWER-UP RULES</td>
<td>SYSTEM RESET</td>
</tr>
<tr>
<td></td>
<td>RUN MM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Executed as soon as you type them and press RETURN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: If you don’t understand everything on this page, stop! Go back over this session until you understand it thoroughly!

MM = Any line number (e.g., 10, 20, 30, etc.)
Math Operators

= (Equal)
+ (Add)
− (Subtract)
* (Multiply)
/ (Divide)
\ (Exponentiation)

(\) means raising a number to a power like $2^2, 2^3$, or $2^4$
Order of Arithmetic Operations

- Multiply → Divide → Add → Subtract
  (Left to Right)
- "My Dear Aunt Sally"

- If Parentheses are used
  - Innermost level operations first
  - Then next level out
  - M.D.A.S. order inside parentheses
Order of Operations Example — (Without Parentheses)

• If there are no parentheses, the computer performs operations by going from left to right doing exponentiation operations (\(^\)\) first. Then (\(*\)) and (\(^/\)) are done in order from left to right and finally (\(+\)) and (\(-\)) are done in order from left to right. (Remember M.D.A.S.!)  

Example:

\[
4 + 5 \times 4^3 - 4/2 = \\
4 + 5 \times \underline{64} - 4/2 = \\
4 + \underline{320} - 4/2 = \\
4 + \underline{320} - \underline{2} = \\
\underline{324} - 2 = \underline{322}
\]
Order of Operations Example — (With Parentheses)

- If there are parentheses, the computer starts at the inner pair of parentheses and converts everything to a single number. Then the computer repeats the process with the next pair of parentheses working “inside” out.

Example:

\[( (6 + 4) \times 2) / 4 = \]

\[( 10 \times 2) / 4 = \]

\[20 / 4 = 5 \]

\[10 \times 2 = 20 \]

\[6 + 4 = 10 \]
In-Class Exercise 4-1

You Try Some Now (Without Parentheses)

1) \(2 \wedge 3 + 4 \times 5 - 4/2 \times 5 = \) _____
2) \(14 - 2 \times 2 + 6 - 2 \times 3 \times 2 = \) _____
3) \(14/2 \times 3 - 2 \wedge 3 + 4 = \) _____

Now Try Some With Parentheses

1) \(6 + (9 \times 2) = \) _____
2) \((6 + (9 \times 2)) \times 5 = \) _____
3) \(3 \times ((4 + (6 \times 2)) \times (9/3 - 1)) = \) _____

A computer is not required here, but it could be used to check the answers. You don’t need a line number for calculator mode. Simply type PRINT and the calculations you want done. Example: If you wish to multiply 2 asterisk 3, simply PRINT 2 * 3 and press RETURN. The answer (6) will be displayed.
Tips on Using Parentheses — Summary

• When in doubt, use parentheses. They can't do any harm!
  — Use parentheses around operations you want performed first
• Make sure that every left parenthesis has a matching right parenthesis
  — Count them to be sure!
• Order of Operations
  — Inner most pair of parentheses first (M.D.A.S. rule inside parentheses)
  — Then work “inside” out
  — In case of a “tie,” computer starts to the left and works right doing
    exponentiation (\(\wedge\)) and the M.D.A.S. rule.

• Assignment 4-1
  — Given the formula for converting Fahrenheit to Celsius as follows:
    \[C^\circ = (F^\circ - 32) \times \frac{5}{9}\]
    1. Write and RUN a program that converts 75\(^\circ\) Fahrenheit to Celsius.
    2. Change the value of F from 75\(^\circ\) to 45\(^\circ\) and RUN the program again.
  — Given the formula for converting Celsius to Fahrenheit as follows:
    \[F^\circ = \frac{9}{5} \times C^\circ + 32\]
    1. Write and RUN a program to find F if C is 20\(^\circ\).
    2. Change the value of C from 20\(^\circ\) to 35\(^\circ\) and RUN the program to find F.
Variable Names Used with Atari BASICA

• Must begin with a capital letter (A-Z)
  — The rest of the characters used in naming a variable can be capital letters or digits
  — The maximum length of a variable name is 114 characters
• Some examples of variable names include:
  — A, B, A7, C302, AAB, AMOUNTCHARGED
• Some examples of illegal variable names include:
  — 7AB, Total, PERSON'SADDRESS, PLACE BORN
• There are some words with special meaning in the BASIC language and they cannot be used as variable names.
  — The complete list of reserved words, which cannot be used in variable names, appears in Appendix A of the Atari BASIC Reference Manual.
In-Class Exercise 4-2
(Assigning Numeric Values to Variables)

10 LET A = 12
20 LET B = 8
30 LET C = A + B
40 LET D = A - B
50 LET E = A * B
60 PRINT A; B; C; D; E
70 LET A = A * 10
80 LET B = A + B
90 LET W = A + B
100 PRINT W
110 END
RUN

<table>
<thead>
<tr>
<th>READY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 LET A = 12</td>
</tr>
<tr>
<td>20 LET B = 8</td>
</tr>
<tr>
<td>30 LET C = A + B</td>
</tr>
<tr>
<td>40 LET D = A - B</td>
</tr>
<tr>
<td>50 LET E = A * B</td>
</tr>
<tr>
<td>60 PRINT A; B; C; D; E</td>
</tr>
<tr>
<td>70 LET A = A * 10</td>
</tr>
<tr>
<td>80 LET B = A + B</td>
</tr>
<tr>
<td>90 LET W = A + B</td>
</tr>
<tr>
<td>100 PRINT W</td>
</tr>
<tr>
<td>110 END</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

~
### Basic Program for a Mathematical Operation

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Key Word</th>
<th>Other Part of Statement</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>LET</td>
<td>X = 5</td>
<td>RETURN</td>
</tr>
<tr>
<td>20</td>
<td>LET</td>
<td>Y = 12</td>
<td>RETURN</td>
</tr>
<tr>
<td>30</td>
<td>LET</td>
<td>Z = X*Y</td>
<td>RETURN</td>
</tr>
<tr>
<td>40</td>
<td>PRINT</td>
<td>Z</td>
<td>RETURN</td>
</tr>
<tr>
<td>99</td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RUN

(1) LET is an optional key word for Atari BASIC. Some computers require you to use LET however. Beware of this if you use another computer.
### Analysis of the BASIC Program for a Mathematical Operation

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Statement</th>
<th>Meaning to Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>LET X = 5</td>
<td>Assign a value of 5 to variable X</td>
</tr>
<tr>
<td>20</td>
<td>LET Y = 12</td>
<td>Assign a value of 12 to variable Y</td>
</tr>
<tr>
<td>30</td>
<td>LET Z = X*Y</td>
<td>Take the values of X and Y, multiply them together, and assign the resulting value to the variable Z</td>
</tr>
<tr>
<td>40</td>
<td>PRINT Z</td>
<td>Print the value of Z (which is 60 in the example)</td>
</tr>
<tr>
<td>99</td>
<td>END</td>
<td>END PROGRAM</td>
</tr>
<tr>
<td>RUN</td>
<td></td>
<td>EXECUTE PROGRAM</td>
</tr>
</tbody>
</table>
# A BASIC Mathematical Program — Area of Rectangle

## Your Action | Display
---|---
1. Type NEW and press RETURN. | 5 PRINT CHR$(125)
2. Type and enter. Line 5 clears the screen. | 10 REM AREA OF A RECTANGLE PROBLEM
 | 20 REM AREA (A) = LENGTH (L) * WIDTH (W)
 | 30 LET L = 10
 | 40 LET W = 5
 | 50 LET A = L * W
 | 60 PRINT A
 | RUN ▶
 | 50
 | READY

## Notes:
- **A** In Line 60, there were no quotes around the letter A because we wanted the computer to PRINT the value of A. If we wanted the computer to PRINT the exact word or letter, we would put quotes around the word or variable.
- **B** After pressing RETURN, the screen clears before the answer is printed.
Area of Rectangle Program Modified

YOUR ACTION | DISPLAY
---|---
1. Type 5, then press **RETURN**. | 70 PRINT "AREA (IN SQ. IN.) IS", A
2. Add Line 70 to read then press **RETURN**. | AREA (IN SQ. IN.) IS 50
3. Type RUN and press **RETURN**. | 80 PRINT "THE AREA IS", A, "SQ. IN."
4. Add Line 80 to read then press **RETURN**. | THE AREA IS 50 SQ. IN.
5. Type RUN and press **RETURN**. | 90 PRINT "THE AREA IS"; A; "SQ. IN."
6. Add Line 90 to read then press **RETURN**. | 100 PRINT "THE AREA IS","A","SQ. IN."
7. Type RUN and press **RETURN**. | THE AREA IS 50 SQ. IN.

Notes:

**A** Comma in Line 70 told the computer to print two separate items on the same line.

**B** Commas in Line 80 told the computer to print three separate items on the same line.

**C** In Line 90, a semicolon tells the computer to print the output close together without spacing. But in line 100, we inserted a space between the word "is" and the second quotes ("). Also, we inserted a space between the third quote and the word "sq." Note the difference in the outputs.

**D** LIST your program when you finish. Run your program several times and note that you have printed your answer five different ways.
Assignment 4-1

1. Write a Program to Find Area of a Triangle
   A. GIVEN: \( A = \frac{1}{2} bh \) WHERE \( b = 5, h = 10 \)
   B. Include Remarks Statement
   C. Have Program Print “THE AREA = ” (Your Answer) “SQ. FT.”

2. Write a Program to Find the Volume of a Rectangular Solid
   A. GIVEN \( V = L \times W \times H \), \( L = 5, W = 10, H = 2 \)
   B. Include Remarks Statement
   C. Have Program PRINT “THE VOLUME = ” (Your Answer) “CUBIC IN.”
Summary — Mathematical Operations

• LET is an optional key word when using Atari BASIC.
  — Other computers using BASIC might require use of LET, so beware!

• 10 PRINT A: Tells computer to print the value of A
  — Whereas 10 PRINT “A”: Tells computer to print letter A (because the computer will print anything within quotes).

• A comma in a PRINT statement tells the computer to leave several spaces between items separated by the commas.

• A semicolon in a PRINT statement tells the computer to print the output close together without spacing.
The Atari is divided into four PRINT zones.
- Each of the first three PRINT zones has 10 spaces for up to 10 characters.
- The fourth PRINT zone has 8 spaces for up to 8 characters.
- The Atari can display up to 38 characters per line ( \(3 \times 10 + 8 = 38\)).
- There are 2 spaces at the beginning of a line which are not used when LISTing or RUNning most programs. (You will learn how to place characters in the first 2 spaces later on.)
Print Zones and the Use of Commas

**YOUR ACTION**

1. Type **NEW** and press **RETURN**.
2. Type **Line 10 to read** then press **RETURN**.
3. Type **RUN** and press **RETURN**.
4. Type **Line 20 to read** then press **RETURN**.
5. Type **RUN** and press **RETURN**.

**DISPLAY**

10 PRINT "ZONE 1", "ZONE 2", "ZONE 3", "ZONE 4"

<table>
<thead>
<tr>
<th>Display</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZONE 1</td>
<td>ZONE 2</td>
</tr>
<tr>
<td>ZONE 1</td>
<td>ZONE 2</td>
</tr>
<tr>
<td>ZONE 1</td>
<td>ZONE 3</td>
</tr>
</tbody>
</table>

**NOTES**

A There are three (3) 10-character PRINT zones and one 8-character PRINT zone per line (since $3 \times 10 + 8 = 38$, the screen can display up to 38 characters per line).

B Note that there are two commas between **ZONE 1** and **ZONE 3**.

C The comma tells the computer to move to the next PRINT zone each time a comma is encountered.
Semicolon vs. Comma

YOUR ACTION
1. Type NEW and press [RETURN].
2. Type exactly as shown
   then [RETURN].
3. Type exactly as shown
   then [RETURN].
4. Type RUN and press [RETURN].
5. Type Lines 30, 40, 50, and 60 as shown
   then press [RETURN].
6. Type RUN 30 and press [RETURN].

THE DISPLAY READS:

10 PRINT "A";"SEMICOLON";"PACKS";"ITEMS";"CLOSE";"TOGETHER"
20 PRINT "BUT A COMMA";"LEAVES";"SPACE S"

ASEMICOLONPACKSITEMSCLOSETOGETHER
   BUT A COMMA   LEAVES   SPACES

30 LET A = 5
40 LET B = 10
50 LET C = 15
60 PRINT A; B; C
   51015
Use of the Semicolon — Summary

- The effect of the semicolon from computer to computer varies, but it is always true that a semicolon leaves less space between the answers or results printed than the COMMA.
- GENERAL RULE: when you want more than one item on the same line and
  - If you want your results or output spread out, use a comma.
  - If you want your results or output close together, use a semicolon.
PRACTICE 6

Area of a Rectangle Program

Part I
1. Enter and RUN this program:
   10 REM AREA OF A RECTANGLE PROGRAM
   20 REM AREA (A) = LENGTH(L) * WIDTH(W)
   30 LET L = 10
   40 LET W = 5
   50 LET A = L * W
   60 PRINT A
2. Add a new program line to include a label on your answer. For example, the area of the rectangle is 50 square inches.
3. Add new program lines to PRINT the following:
   a. The length of the rectangle is 10 inches.
   b. The width of the rectangle is 5 inches.

Part II
1. Do not type NEW.
2. Change the values of L and W in the program. (Think before you change the lines! How many lines do you have to change? Change only those lines!)

PRACTICE 7

Program Using Mathematical Operators

1. Enter and RUN the following program:
   10 REM MATH PROBLEMS
   20 LET A = 75
   30 LET B = 50
   40 LET C = A + B
   50 PRINT C
2. Change the values of A and B in the program and RUN it. Fill in the results: A = ______, B = ______, C = ______.
3. Add a program line to label the answer. Example: “The sum is (your answer).”
4. Write a program to multiply (*) two numbers (any two).
5. Add the program line to PRINT: “The product of (your no.) “*” (your no.) “is” (your answer).
   Example: The product of 5 * 5 is 25.
6. Write a program to divide (/) two numbers (any two).
7. Add the program line to PRINT: “The quotient of” (your #) “/” (your #) “is” (your answer).
   Example: The quotient of 10/2 is 5.
8. Write a program to subtract (−) two numbers (any two).
9. Add the program line to PRINT: “The difference between “(your #) “−” (your #) “is” (your answer).
   Example: The difference between 10−5 is 5.

Additional practices for this Part will be found in the back of the book.
PART 5

Scientific Notation

What You Will Learn

To understand and use scientific notation.

Review and Feedback

The purpose of this part of the program is to evaluate students' overall performance and determine which students are having problems. The students who are having problems will be given the opportunity to review concepts they have not mastered. The review and feedback phase is divided into the following parts:

1. Exam — written/lab
2. Open discussion with students about their concerns and interests
3. Evaluation of student's performance
4. Recommendations
Scientific Notation

• Scientists often express large numbers like 186,000 and small numbers like 0.00015 as the product of two numbers. For example:

a) \(186,000 = 1.86 \times 10^5\)
b) \(0.00015 = 1.5 \times 10^{-4}\)
c) \(764,000 = 7.64 \times 10^5\)
d) \(0.0347 = 3.47 \times 10^{-2}\)
e) \(5,000,000 = 5 \times 10^6\)
# Scientific Notation

<table>
<thead>
<tr>
<th>Ordinary Notation</th>
<th>Scientific Notation</th>
<th>Scientific Notation in Atari BASIC</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000,000,000</td>
<td>5 X 10^9</td>
<td>5E + 09</td>
<td>ADD 9 zeroes after 5</td>
</tr>
<tr>
<td>.000005</td>
<td>5 X 10^-6</td>
<td>5E - 06</td>
<td>Shift decimal 6 places to left</td>
</tr>
<tr>
<td>.00000005</td>
<td>5 X 10^-8</td>
<td>5E - 08</td>
<td>Shift decimal 8 places left</td>
</tr>
<tr>
<td>5 (with 15 zeroes)</td>
<td>5 X 10^15</td>
<td>5E + 15</td>
<td>ADD 15 zeroes after 5</td>
</tr>
<tr>
<td>5 (with 16 zeroes)</td>
<td>5 X 10^16</td>
<td>5E + 16</td>
<td>ADD 16 zeroes after 5</td>
</tr>
</tbody>
</table>

- Atari BASIC uses scientific notation for very large and very small numbers.
- Rule 1: E + 09 means move the decimal point 9 places to the right.
- Rule 2: E - 09 means move the decimal point 9 places to the left.
Assignment 5-1 — (Scientific Notation)

1. Type, enter, and RUN the following program:

   5 PRINT CHR$(125)
   10 PRINT 5000 000, 0.000005, .00000005, 5 000 000 000
   15 PRINT
   20 PRINT 5 000 000 000 000 000, 5 000 000 000 000 000
       (15 zeroes)             (16 zeroes)

2. Experiment with scientific notation until you feel comfortable with it.
Review and Feedback

A. Quiz — Written/Lab
B. Open discussion with students on concerns and interest
C. Evaluation of student’s performance
D. Recommendations
FEEDBACK QUESTIONNAIRE

1. Do you like working with computers? yes, no. If not, why not? __________
2. What things do you like most about computers? __________
3. What do you dislike most about computers? __________
4. If you were a design engineer and could design the computer to do anything you wanted it to, what kinds of things would you include in your design? (Use your imagination!)
   __________
5. What was the hardest thing for you to understand about the computer so far? __________
6. What was the easiest thing for you to understand? __________
7. Were you afraid or nervous when you first used the computer? yes, no
8. Do you feel comfortable using the computer now? yes, no
9. Would you prefer to be doing something else rather than learning about computers? yes, no. If yes, what would you like to do? __________
10. Is the teacher going too fast, too slow, or just right for you? __________
11. Do you find the lessons interesting, boring, or so-so? __________
12. If you could teach this course, what would you do to make the lessons more interesting? __________
13. Have you decided what you want to do for a vocation? yes, no
   If yes, what? __________
14. Would you like to take additional courses to learn more about computers and programming? yes, no
15. Do you have any additional comments? __________
PRACTICE 8

Scientific Notation

1. Convert the following to standard scientific notation (example: \(5,000,000 = 5 \times 10^6\)):
   a. 5,165,123
   b. .000007
   c. .00000008
   d. 6,001,255
   e. 80 000 000 000 000 000 (16 zeros)
   f. 8000 000 000 000 000 (15 zeros)
   g. 9,000,156,000
   h. 7,701,777
   i. 77,701,777,000
   j. 5,612,345,000

2. Change the above numbers to computer scientific notation used in the Atari (example: \(5,000,000,000 = 5E+09\)).

Note: The Atari will print a number in scientific notation if:
   A. For positive numbers
      1. The value is greater than 999999999
      2. The value is less than .01
   B. For negative numbers
      1. The value is less than -999999999
      2. The value is greater than -.01

Another way of indicating this is to say that the number will be printed in scientific notation if its absolute value is larger than 999999999 or less than .01.
PART 6

Relational Operators and IF-THEN/GOTO Statements

What You Will Learn

1. How computers compare (or relate) one value with another.
2. To explain the purpose and use of the six relational operators: =, >, <, <=, >=, <>
3. To explain the purpose and use of the key words IF-THEN, GOTO.
4. To write, enter, and run programs that use IF-THEN and GOTO statements.
5. To understand and use the counting program.
Relational Operators

- Allow computer to compare one value with another.
  - The three relational operators include

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal</td>
<td>A = B</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>A &gt; B</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>A &lt; B</td>
</tr>
</tbody>
</table>

- Combining the three operators above we have

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&gt;</td>
<td>Is not equal to</td>
<td>A &lt;&gt; B</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>A &lt;= B</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>A &gt;= B</td>
</tr>
</tbody>
</table>

NOTE: To distinguish between < and >, just remember that the smaller part of the < symbol points to the smaller of two quantities being compared.
• IF-THEN is used in conditional branching.
  — That is, the program will "branch" to another part of the program on the condition that it passes the test it contains.
  — If the test fails, the program simply continues to the next line.

• Example:

<table>
<thead>
<tr>
<th>LINE NUMBER</th>
<th>KEY WORD</th>
<th>CONDITION TO BE TESTED</th>
<th>KEY WORD</th>
<th>IF &quot;YES&quot; BRANCH TO LINE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>IF</td>
<td>A = 5</td>
<td>THEN</td>
<td>50</td>
</tr>
<tr>
<td>30</td>
<td>IF &quot;NO&quot;</td>
<td>NEXT LINE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sample Program Using IF-THEN (Conditional Branching)

• Program
  10 LET A = 5
  20 IF A = 5 THEN 50
  30 PRINT "A DOES NOT EQUAL 5"
  40 END
  50 PRINT "A EQUALS 5"
  RUN

• The screen should display
  A EQUALS 5

• Why is Line 20 above a conditional branching statement?
  — What's the condition or test?
In-Class Exercise 6-1 (IF-THEN)

Given: \( A = 10, B = 20, C = 30 \)

Exercises:

<table>
<thead>
<tr>
<th>Exercise No.</th>
<th>Statement</th>
<th>Condition is (T or F)</th>
<th>Branch to (Line N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>10 IF ( A = B ) THEN 40</td>
<td>F</td>
<td>20</td>
</tr>
<tr>
<td>2.</td>
<td>10 IF ( A &lt;&gt; B ) THEN 50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>10 IF ( A &gt; B ) THEN 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>10 IF ( A &lt; B ) THEN 70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>10 IF ( C &lt;= A + B ) THEN 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>10 IF ( C &gt;= A + B ) THEN 90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>10 IF ( B &gt; A ) THEN 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>10 IF ( B/A &gt;= C/A ) THEN 110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>10 IF ( A * B &lt;= A * C ) THEN 120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>10 IF ( C/A &lt;= A * B ) THEN 130</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \text{A} \) Note: If condition is false (F), the computer will execute the next line (i.e., 20).
A Counting Program — Using IF-THEN

• Program
  10 LET J = 0
  20 LET J = J + 1
  30 PRINT J
  40 IF J < 10 THEN 20
  RUN
  OUTPUT IS*

• In-Class Exercise 6-2
  Modify above program to count to 50 by 5's

* OUTPUT
  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
**IF-THEN Counter Program Analysis**

<table>
<thead>
<tr>
<th>PROGRAM EXECUTION</th>
<th>&quot;J&quot; COUNTER STATUS</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE</td>
<td>1stä J = 0</td>
<td>ø</td>
</tr>
<tr>
<td>1ST TIME</td>
<td>2stä J = J + 1</td>
<td>1 = ø + 1</td>
</tr>
<tr>
<td></td>
<td>3stä PRINT J,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4stä IF J &lt; 4 THEN 2stä</td>
<td></td>
</tr>
<tr>
<td>2ND TIME</td>
<td>2stä J = J + 1</td>
<td>2 = 1 + 1</td>
</tr>
<tr>
<td></td>
<td>3stä PRINT J,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4stä IF J &lt; 4 THEN 2stä</td>
<td></td>
</tr>
<tr>
<td>3RD TIME</td>
<td>2stä J = J + 1</td>
<td>3 = 2 + 1</td>
</tr>
<tr>
<td></td>
<td>3stä PRINT J,</td>
<td></td>
</tr>
<tr>
<td>4TH TIME</td>
<td>4stä IF J &lt; 4 THEN 2stä</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>5stä END</td>
<td></td>
</tr>
</tbody>
</table>

1 2 3 4
### IF-THEN COUNTER — Program Analysis (Stop-Action)

<table>
<thead>
<tr>
<th></th>
<th>PROGRAM EXECUTION</th>
<th>“J” COUNTER STATUS</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIALIZE</td>
<td>( 10 \ J = 0 )</td>
<td>( 10 \ 0 )</td>
<td>1</td>
</tr>
<tr>
<td>1ST TIME</td>
<td>( 20 \ J = J + 1 )</td>
<td>( 20 \ 1 = 0 + 1 )</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>( 30 ) PRINT J</td>
<td>( 30 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 40 ) STOP</td>
<td>( 40 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 45 ) REM TYPE CONT TO CONTINUE</td>
<td>( 45 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 50 ) IF ( J &lt; 4 ) THEN ( 20 )</td>
<td>( 50 )</td>
<td></td>
</tr>
<tr>
<td>2ND TIME</td>
<td>( 20 \ J = J + 1 )</td>
<td>( 20 \ 2 = 1 + 1 )</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>( 30 ) PRINT J</td>
<td>( 30 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 40 ) STOP</td>
<td>( 40 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 45 ) REM TYPE CONT TO CONTINUE</td>
<td>( 45 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 50 ) IF ( J &lt; 4 ) THEN ( 20 )</td>
<td>( 50 )</td>
<td></td>
</tr>
<tr>
<td>3RD TIME</td>
<td>( 20 \ J = J + 1 )</td>
<td>( 20 \ 3 = 2 + 1 )</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>( 30 ) PRINT J</td>
<td>( 30 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 40 ) STOP</td>
<td>( 40 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 45 ) REM TYPE CONT TO CONTINUE</td>
<td>( 45 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 50 ) IF ( J &lt; 4 ) THEN ( 20 )</td>
<td>( 50 )</td>
<td></td>
</tr>
<tr>
<td>4TH TIME</td>
<td>( 20 \ J = J + 1 )</td>
<td>( 20 \ 4 = 3 + 1 )</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>( 30 ) PRINT J</td>
<td>( 30 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 40 ) STOP</td>
<td>( 40 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 45 ) REM TYPE CONT TO CONTINUE</td>
<td>( 45 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( 50 ) IF ( J &lt; 4 ) THEN ( 20 )</td>
<td>( 50 )</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>( 60 ) END</td>
<td>( 60 )</td>
<td></td>
</tr>
</tbody>
</table>
In-Class Exercise 6-3
(GOTO — Unconditional Branching)

• Type and RUN this program:
  10 PRINT CHR$(125)
  20 PRINT "YOUR NAME";
  30 GOTO 20

• What happened?
  — Do you know how to stop the program? (What about the \textbf{BREAK} key?)
  Explain this simple program (Line 10 merely clears the screen).
  But what does Line 30 tell the computer to do?
  — Were there any tests or conditions to be satisfied
    in Line 30 before it does what it has to do?
  — Do you understand now why the GOTO statement is called an
    unconditional branching statement?

• Don’t leave this page until you understand everything!
Exercise 6-4 (GOTO/IF-THEN)

Exercise:
• Study the program below and write the message that would be printed if the program were executed.

10 PRINT "WELCOME TO LEEDS MIDDLE SCHOOL"
20 GOTO 70
25 PRINT
30 PRINT "HELLO SUPERSTAR"
35 PRINT
40 PRINT "COMPUTERS ARE MY THING"
50 GOTO 100
60 IF A = 5 THEN 90
70 PRINT "COMPUTER WORKSHOP"
80 GOTO 40
90 GOTO 120
100 LET A = 5
110 GOTO 60
120 PRINT "AND I'M A SUPERSTAR!"
130 END
140 PRINT "ATARI MICROCOMPUTER"
150 PRINT "I CAN DO IT TOO"
160 PRINT "I SPEAK BASIC"
Assignment 6-1

1. Read pages 18 and 19 in the *Atari BASIC Reference Manual*.

2. Write a program of your choice using conditional (IF-THEN) and unconditional (GOTO) statements.

3. Write a counting program.
   — Count to 100 by 10's.
What We Have Learned — Summary

- Relational operators: =, >, <, <=, >=
- IF-THEN
- GOTO (No space between GO and TO)
- Conditional Branching
  - If condition is met, (i.e., TRUE), branch to designated line in program.
  - If condition is not met, (i.e., FALSE), go to next line number in program.
- Unconditional branching
  - GOTO line XX (no conditions or tests required)
  - A GOTO statement, as the name implies, forces the computer to go to a specific statement anywhere in the program.
PRACTICE 9

Using IF-THEN

Part I.
1. Enter and RUN the following program:
   10 LET A = 10
   20 IF A = 10 THEN 50
   30 PRINT "A DOES NOT EQUAL 10"
   40 END
   50 PRINT "A EQUALS 10"
2. Change Line 10 to Let A = 5 and then RUN it.
3. Change Line 10 to Let A = 3 and then RUN it.

Part II.
1. Using this program as an example, write a new program to PRINT A EQUALS 3 and RUN it.
2. Change the values of A in Line 10 and RUN the program several times.

PRACTICE 10

Counting Program Using IF-THEN

1. Enter and RUN this program:
   10 LET J = 0
   20 LET J = J + 1
   30 PRINT J
   40 IF J < 10 THEN 20
2. Write a program to count from 1 to 15.
3. Write a program to count to 50 by 5's.
4. Write a program to count to 100 by 10's.
5. Write a program to count from 15 to 30 and PRINT the answers in one column (vertically).
   Example: 15
   16
   17
   18
   and so forth
6. Write a program to count from 20 to 40. PRINT answers horizontally, with one space between answers.
   Example: 20 21 22 23 and so forth.
PART 7

Input Statements

What You Will Learn

1. To explain the purpose and use of the key word input.
2. To explain the purpose and use of a trailing semicolon on a program line.
3. To identify and use string variables A$, B$, C$, and so forth.
4. To explain the difference between numeric and string variables.
5. To write, enter, and run programs that use the concepts of this lesson.
Input Statement

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 INPUT A</td>
<td>• Causes the computer to stop, PRINT a ?, and wait for you to type in a decimal number.</td>
</tr>
<tr>
<td></td>
<td>• After you type in a value for A, the computer continues the program when you press the [RETURN] key.</td>
</tr>
</tbody>
</table>
Input Statements

YOUR ACTION
1. Type NEW and press RETURN.
2. Type and enter Lines 5 & 10 as shown.
3. Type RUN and press RETURN.
4. Enter a number (e.g., type 5 and enter).
5. RUN this program several times to get the feel of it.

DISPLAY
5 PRINT "THE # I'M THINKING OF IS"
10 INPUT A
THE # I'M THINKING OF IS
? A
THE # I'M THINKING OF IS
? 5
READY

A The question mark on the screen means, “It's your turn and I'm waiting.”
Input Statements —
(Area of Rectangle Program)

10 REM AREA OF A RECTANGLE PROBLEM
20 REM A = L * W
30 PRINT "THE LENGTH IS"
40 INPUT L
50 PRINT "THE WIDTH IS"
60 INPUT W
70 A = L * W
80 PRINT "THE AREA IS"
90 PRINT A
Area of Rectangle Problem Revisited
(Using Input Statements)

YOUR ACTION | DISPLAY
1. Type in program Lines 10 through 60 as shown.

2. Type RUN then press RETURN.
3. Type in the length (say 10) and enter.
4. Type in the width and press RETURN.
5. What is your answer?

A Note the trailing semicolon. It is used to hook Lines 50 and 60 together.

B Note that the program waits for an input from the keyboard. If you don't enter a number or press RETURN, it will just stay at that line until the machine is turned off or reset.
Assignment 7-1

Write a simple program to do the following:
(using input statement)
a) Input your age
b) Input your zip code
c) Input your weight
d) Input your height in inches
e) PRINT each of the above with the proper labels
   (for example: My age is 15 or I am 15 years old).
What We Have Learned

- Trailing semicolon hooks two lines together.
- Input statements cause the computer to stop and wait for an input from the keyboard.
- Input statements cause the Atari to print a ?.
### Numeric vs. String Variables

<table>
<thead>
<tr>
<th>Numeric Variable</th>
<th>Declaration Character</th>
<th>String Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$</td>
<td>A$</td>
</tr>
<tr>
<td>A1</td>
<td>$</td>
<td>A1$</td>
</tr>
<tr>
<td>CHECKBAL</td>
<td>$</td>
<td>CHECKBAL$</td>
</tr>
<tr>
<td>ANDMORE</td>
<td>$</td>
<td>ANDMORE$</td>
</tr>
</tbody>
</table>

### The DIM Statement

In order to use a string variable with the Atari, it is necessary to tell the Atari the maximum number of characters that can be stored in the variable. This is accomplished with the command DIM. For example, DIM A$(6) tells the Atari to leave space for a maximum of 6 characters to be stored in the string variable A$. DIM stands for DIMension.

**Note:**

1. If you accidentally try to assign a string with more than 6 characters to A$, the Atari will simply ignore all characters beginning with the seventh.
2. Once you DIM a string variable, you cannot DIM it to a different length (unless you type NEW first, which would result in the loss of your program).
Example of Use of String Variables

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
</table>
| 1. Type and enter. | 10 PRINT CHR$(125)  
20 DIM A$(20)  
30 PRINT “YOUR NAME IS”;  
40 INPUT A$  
50 PRINT “HELLO THERE, “; A$ |
| 2. Type RUN and press RETURN. | YOUR NAME IS? ■  
HELLO THERE, BILL  
READY  
 ■ |
| 3. Type RUN and press RETURN | YOUR NAME IS? |
| 4. Type and enter. | YOUR NAME IS? WILLIAM THE FAMOUS SUPER STAR  
HELLO THERE, WILLIAM THE FAMOUS S  
READY  
 ■ |

**A** NOTE:  
It will print your name and not “BILL,” unless your name is “BILL.”

**B** Since you entered more than 20 characters (remember, spaces count, too), the Atari stored only the first 20 characters in A$.
In-Class Exercise 7-1 (String Variables)

YOUR ACTION | DISPLAY
-------------|-------------------
1. Type and enter. | 5 PRINT CHR$(125)  
10 DIM A$(10), B$(10), C$(20)  
20 PRINT "YOUR FIRST NAME";  
30 INPUT A$  
40 PRINT "YOUR MIDDLE NAME";  
50 INPUT B$  
60 PRINT "YOUR LAST NAME";  
70 INPUT C$  
80 PRINT A$; " "; B$; " "; C$  

2. Type RUN and press [RETURN]. | YOUR FIRST NAME? AUBREY  
YOUR MIDDLE NAME? BRIGHT  
YOUR LAST NAME? JONES  
AUBREY BRIGHT JONES  
READY

NOTES
You can combine string variables.  
You must insert a space between string variables using " " marks.  
A semicolon will not cause a space to be printed.
Assignment 7-2 (String Variables)

1. Run and analyze the following program:

```
10 DIM A$(25), B$(30)
20 PRINT "YOUR NAME IS";
30 INPUT A$
40 PRINT "YOUR HOUSE NUMBER";
50 INPUT A
60 PRINT "YOUR STREET NAME";
70 INPUT B$
80 INPUT "YOUR ZIP CODE";
90 INPUT B
100 PRINT A$
110 PRINT A; " "; B$
120 PRINT "ZIP CODE " ; B
```

2. Answer the following questions:
   a) Why were A$ and B$ (string variables) required in Lines 30 and 70?
   b) Why were quotes (" ") inserted in Line 110?
   c) Why did we use $ symbol (or string declaration character) with A and B in Lines 30 and 70?
String Variables — Summary

- String variables can be assigned to indicate letters, words, and/or combinations of letters.
- String variables can be combined.
- Use " " marks to insert a space between string variables.
PRACTICE 11

Area of Rectangle Problem (Using INPUT Statement)
1. Enter and RUN this program:
   
   10 PRINT "THE LENGTH IS";
   15 INPUT L
   20 PRINT "THE WIDTH IS";
   25 INPUT W
   40 LET A = L*W
   50 PRINT "THE AREA IS "; A

2. Write a new program using INPUT statements to find volume (volume = length × width × height).
3. Include a statement: The volume is ________.

PRACTICE 12

More INPUT Statement Programs

Part I.
1. Write a program using INPUT statements to change meters to centimeters (centimeters = 100 × meters).
2. Include a statement: ________ meters equals ________ centimeters.

Part II.
1. Write a new program using INPUT statements to do the following:
   a. Input your age.
   b. Input your zip code.
   c. Input your weight.
   d. Input your height.
2. PRINT each with the proper labels.
   Example: My age is ________.

PRACTICE 13

String Variables

Part I.
1. Enter and RUN the following program:
   
   10 DIM A$(25), B$(30)
   20 PRINT "YOUR NAME IS" ;
   30 INPUT A$
   40 PRINT "YOUR HOUSE NUMBER" ;
   50 INPUT A
   60 PRINT "YOUR STREET NAME" ;
   70 INPUT B$
   80 PRINT "YOUR ZIP CODE" ;
   90 INPUT B
   100 PRINT A$ ; " "; B$
   110 PRINT A ; " "; B$
   120 PRINT "ZIP CODE "; B

2. Answer the following questions:
   a. Why are A$ and B$ (string variables) required in Lines 30 and 70?
   b. Why were quotes (" ") inserted in Line 110?
   c. Why didn't we use $ symbol (or string declaration character) with A and B in Lines 50 and 90?

Part II.
1. Write a new program using INPUT statements, string variables, and a space between each line, PRINT all information (example: My best friend is ________) to give the following information:
   a. Your best friend.
   b. Your favorite subject.
   c. Your favorite food.
   d. Your favorite movie star.
   e. Your favorite color.
   f. Your zodiac sign.
What You Will Learn

1. To define and use the terms bit, byte, k, kbytes.
2. To determine how much memory is used in a BASIC program.
3. To explain the purpose and use of the command PRINT FRE (0).
4. To use the Atari in calculator mode (i.e., without having to write a program).
BIT = BINARY—DIGIT

BIT = SMALLEST MEMORY CELL IN A COMPUTER

BIT = "1" OR "0"
MEMORY CELL WITH 1 BIT

8 MEMORY CELLS

\[
\begin{array}{ccccccc}
\emptyset & 1 & 1 & \emptyset & 1 & \emptyset & 1 & 1
\end{array}
\]

8 BITS = 1 BYTE
BYTE = 8 BITS
K = 1000
KBYTES = 1000 BYTES
KBYTES = 8000 BITS
## How Much Memory Is Used in BASIC Programs

<table>
<thead>
<tr>
<th>WHAT'S STORED</th>
<th>HOW MUCH MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ALPHA CHARACTER (A-Z)</td>
<td>1 BYTE</td>
</tr>
<tr>
<td>1 SPECIAL CHARACTER (e.g., &quot;!, +, −, etc.)</td>
<td>1 BYTE</td>
</tr>
<tr>
<td>1 NUMERIC CHARACTER (0-9)</td>
<td>1 BYTE</td>
</tr>
<tr>
<td>1 SPACE</td>
<td>2 BYTES</td>
</tr>
<tr>
<td>1 [RETURN] KEY AND OTHER</td>
<td>3 BYTES MEMORY</td>
</tr>
<tr>
<td>1 RESERVE WORD SUCH AS FOR, GOTO, PRINT</td>
<td>1 BYTE OVERHEAD*</td>
</tr>
</tbody>
</table>

**EXAMPLE:**

```
10 PRINT "MY NAME IS AUBREY" RETURN
```

\[2^* + 1^* + 1 + 1^* = 19\]

\[1^* = 25\ \text{BYTES}\]

*Included as part of memory overhead. Memory overhead means you will use 5 bytes of memory for each line, short or long.

**NOTE!** The above is just an exercise to help you understand memory allocation. You don't have to count bytes to determine how much memory was used. (Page 118 will show you an easy way to find out how much memory is available for your use.)
The Memory Command

• PRINT FRE (Ø)
  — This command is used to let you know how much memory is available to you.
  — Sometimes it may be important to know how much memory you are using for a given program.
  — If the amount of memory available in the Atari you are using is 16K, this means that there are about 16,000 different memory locations to store and process your programs (actually 16,384).

• Note!
  — With no program loaded, there are less than 16,384 memory locations available for use. The difference in memory space, between actual space and 16,384, is set aside for processing programs and overall management and monitoring of what the computer is doing.
  — Also, your Atari might have more than 16K of memory. So make certain you know how much memory you have in your computer. (The next page will show you how to determine the amount of memory available to you.)
Assignment 8-1

1. Determining available memory:
   a) Type NEW and press RETURN.
   b) Type PRINT FRE (0) and press RETURN.
   c) Display reads: ________________
   d) Now type the following and enter 10 PRINT
      "Leeds Middle School".
   e) Type PRINT FRE (0) and press RETURN.
   f) How much space is left in memory? ________________

2. Use Atari in calculator or immediate mode to solve
the following:
   a) 25 * 4/2
   b) (25 + 6) − 7 + (2 * 5)
   c) 7/2 * 5 * 2 ∧ 3
   d) Any other problems you want to try

Remember! You don’t need a line number for calculator mode. Simply type PRINT and the calculations you want done. Example: If you wish to multiply 2 asterisk 3, simply type PRINT 2 * 3, and press RETURN. The answer (6) will be displayed.
What We Have Learned

• COMPUTERS SPEAK IN MACHINE LANGUAGE

• MACHINE LANGUAGE IS A FORM OF BINARY CODING

• BINARY CODE CAN BE EITHER "0" OR "1" BITS

• BIT = BINARY DIGIT

• BYTE = 8 BITS

• YOU DO NOT HAVE TO KNOW MACHINE LANGUAGE TO USE COMPUTERS!
PRACTICE 14

Sizing Memory and Calculator Mode

Part I.

1. To determine available memory:
   a. Type NEW and press [RETURN].
   b. Type PRINT FRE (0) and press [RETURN].
   c. Display reads: ____________.
   d. Now type the following and enter 10 PRINT "LEEDS MIDDLE SCHOOL."
   e. Type PRINT FRE (0) and press [RETURN].
   f. How much space is left in memory? ______________

Part II.

1. Use Atari in calculator or immediate mode to solve the following:
   a. 25 * 4/2
   b. (25 + 6) - 7 + (2 * 5)
   c. 7/2 * 5 * 2 \(\wedge\) 3
   d. Any other problems you want to try.
PART 9

Using the Cassette Recorder and the Disk Drive

What You Will Learn

1. How to use the cassette and the disk drive as output devices to save information stored in memory.
2. How to use the cassette and the disk drive as input devices to load information from tape to memory.
3. How to explain and use the commands CSAVE, CLOAD.
A Cassette Recorder Is an I/O Device

Using Cassette Tape Recorder

• The cassette tape recorder is an input/output (I/O) device that allows you to “save” information on cassette or “load” information from cassette.
  — When you have typed a long program and wish to save it, you can save it on cassette (CSAVE).
  — When you are ready to use it again, you can load it from the cassette (CLOAD).
• Note! You can only save your program on cassette (not the program output).
• Refer to the Atari Reference Manual for tips on using the recorder.
Using the Tape Cassette Recorder as an Input Device
(That Is, to Load a Program from Tape Into Memory)

STEP | ACTION
--- | ---
1. | Place a program tape into your recorder.
2. | Rewind tape (if necessary).
3. | Set the volume at approximate level. (You may want to start with volume in the range 5-7). If the volume setting is too low or too high, you will get an error message. To find the right volume setting, you will have to use a trial-and-error method.
4. | Press the PLAY button on the tape recorder.
5. | Type CLOAD and then press RETURN. After one beep, press RETURN again.
6. | It may take up to 15 seconds before one of the following happens:
   a. ERROR 138 appears.
   b. ERROR 143 appears.
   c. READY and the cursor appears.
7. | If case (a) or (b) occurs, rewind tape and try again.
8. | Important Note! Trying to load a tape can be a very frustrating experience sometimes, so hang in there! After you load the tape, move the position of the recorder's volume control so you can use this setting each time you load a tape in the future.
9. | If case (c) above occurs, you did it! (You can RUN your program now.) But be careful! Once in a while the load command may not work properly and although you see the prompt character and the blinking cursor, the tape was not loaded. If this happens, try turning the power off and then on again and then try again to load the tape.
Using the Tape Cassette Recorder as an Output Device
(That Is, to Save a Program from Memory)

STEP    ACTION
1. Place the blank tape in recorder and rewind to the beginning.
2. Advance tape past leader (the nonmagnetic part of the tape).
3. Hold down Play lever while pressing the Record lever on recorder.
4. Type and CSAVE and press RETURN.
5. When you press RETURN, the computer will beep twice, and you should hit RETURN again.
6. When the recording is completed, READY will appear on the screen, with the cursor on the next line.
7. Push the Stop lever on the recorder and rewind the tape. You have saved your program on tape without affecting your program in memory.
The Disk Drive as an I/O Device

- The disk drive is an input/output (I/O) device which allows you to "save" programs on a disk or "load" programs from a disk.

- The disks you will use with the Atari are square pieces of plastic (5 1/4" on a side) which are specially treated so that they can store information from the Atari.

- When you pick up a disk, it is very important that you touch only the disk cover and NEVER TOUCH THE DISK SURFACE (or else the programs on the disk may be destroyed). IF YOU ARE NOT FAMILIAR WITH THE HANDLING OF A DISK, REFER TO PAGE 3 OF THE ATARI MANUAL.

- After you store a program on a disk, you will probably want to write your name on the disk label. Be sure to use a soft-tip pen when writing on the disk label.

- Since the Atari can send and retrieve information to and from a disk at a much faster rate than to and from a tape, whenever possible it's much better to use a disk. However, there are special steps to follow to make sure that your disk is ready to be used. These steps are called INITIALIZING THE DISK.
FORMATTING A DISK

STEP ACTION

1. Be sure the Atari is turned off.
2. Open the disk-drive door by gently pressing the rectangular surface under the door latch.
3. With the drive door open, turn the disk drive on. The drive will whirr for a few seconds.
4. Insert the Atari MASTER DISKETTE (or any Atari diskette that contains Atari DOS* files). Push the diskette into the drive door until you hear a click.
5. Gently close the door by pressing down on the latch until you hear a click.
6. Turn on the Atari. The disk drive will whirr for a few seconds, and you will see:

   READY

7. Type DOS and press [RETURN]. After the drive stops, you will see the DOS menu on the screen.
8. Remove the Atari MASTER DISKETTE. (If necessary, refer to step 2.)
9. Insert a blank diskette. (If necessary, see steps 4 and 5, and be sure you insert a blank diskette.)
10. Press I and [RETURN]. The Atari will ask which drive to format. Press I and [RETURN]. The Atari will ask you to type Y to format the disk. This extra step is to be certain that you have not inserted a disk on which there is information that you want to save. Any information on the disk to be formatted will be lost once it is formatted.
11. Press Y and [RETURN]. Your disk will now be formatted.
12. Press H and [RETURN]. This will copy the Atari DOS files onto your disk.

*DOS = Disk Operating System
Using the Disk Drive as an Output Device
(That is, saving a program on disk)

Actually, in a way you’ve already saved two programs on disk: the two Atari DOS files. Now you will learn how to save other files as well.

STEP ACTION
1. Boot* your disk. (If necessary, see steps 1-6, page 126.)
2. Type NEW.
3. Enter the following program.
   
   ```plaintext
   5 PRINT CHR$(125)  
   10 REM THIS IS MY FIRST PROGRAM  
   20 PRINT "I KNOW HOW TO FORMAT A DISK"  
   30 PRINT  
   40 PRINT "AFTER A DISK IS FORMATTED"  
   50 PRINT  
   60 PRINT "THE ATARI DOS FILES SHOULD"  
   70 PRINT  
   80 PRINT "BE TRANSFERRED TO THE DISK"
   ```
4. Type SAVE "D:MYFILE1" and press [RETURN]. Your program will now be saved.
5. Type DOS and press [RETURN] to display the DOS menu.
6. Press A and press [RETURN] twice. You will see a listing of all the files on your disk. The listing will include your program file along with the two DOS files.

*Boot refers to starting up the Atari as explained in steps 1-6 on page 126.
Using the Disk Drive as an Input Device
(That is, loading a program from disk)

STEP ACTION

1. Boot your formatted disk. (If necessary, see steps 1-6, page 126.)

2. Type LOAD "D:MYFILE1" and press [RETURN]. (Be sure to use the quotes and the colon.)

3. Type LIST and press [RETURN]. You will see the listing of your program.

4. Type RUN and press [RETURN]. You will see:

I KNOW HOW TO FORMAT A DISK
AFTER A DISK IS FORMATTED
THE ATARI DOS FILES SHOULD
BE TRANSFERRED TO THE DISK

READY

Note: You can RUN your program without using the LOAD step by doing the following in place of steps 2, 3, and 4.

Type RUN "D:MYFILE1". This will automatically LOAD and RUN your program.
PRACTICE 15

Using the Computer to Solve Problems
1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer (output).
   The total enrollment at Armstrong High School is 1,264. There are 367 freshmen, 322 sophomores, and 298 juniors. How many seniors are there?
2. Write a new program using INPUT statements to solve one of the problems.

PRACTICE 16

Finding the Average Problems
1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer.
   The weights of three boys are 140 lb, 150 lb, and 130 lb. What is their average weight?
2. Write a new program using INPUT statements to solve the same problem. (That is, you should use the INPUT statement for the weight of the three boys.)

PRACTICE 17

Using the Computer to Solve Problems
1. Write two programs to solve the following problems. Label your answers.
2. Over a period of six years Mr. Smith drove his car 53,862 miles. What was the average distance each year?
3. After 12 dozen bulbs were sold, how many of the 1,000 bulbs were left?
PART 10

Using FOR-NEXT STEP Statements

What You Will Learn

1. To explain the purpose and use of key words FOR-NEXT STEP.
2. To explain the purpose and use of the terms increment, decrement, initialize.
3. To compare key words GOTO, IF-THEN, FOR-NEXT and explain how they relate to one another.
4. To explain the purpose and use of timer loops.
For-Next Statement

• Allows the computer to do the same thing over and over a large number of times (and do it very fast!)
FOR - NEXT Loop

YOUR ACTION

1. Type and enter program as shown.

2. Type RUN and press RETURN.

DISPLAY

5 PRINT CHR$(125)
10 FOR J = 1 TO 10
20 PRINT " AUBREY " ; J
30 NEXT J

AUBREY 1
AUBREY 2
AUBREY 3
AUBREY 4
AUBREY 5
AUBREY 6
AUBREY 7
AUBREY 8
AUBREY 9
AUBREY 10

READY
### FOR-NEXT STEP Loop

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Retype and enter Line 10 of resident* program as shown.</td>
<td>10 FOR J = 1 TO 10 STEP 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUBREY 1</td>
</tr>
<tr>
<td>AUBREY 4</td>
</tr>
<tr>
<td>AUBREY 7</td>
</tr>
<tr>
<td>AUBREY 10</td>
</tr>
</tbody>
</table>

READY

*Resident means program currently in memory.

**A** If step is not included in the statement, an increment of 1 is assigned by the computer (i.e., step 1).
Example of Program Statements Using Key Words

FOR-NEXT STEP

\[ \begin{align*}
10 & \quad \text{FOR J = 10 TO 1 STEP -1} \\
20 & \quad \text{PRINT J; " ";} \\
30 & \quad \text{NEXT J}
\end{align*} \]

RUN
DISPLAY READS:
10 9 8 7 6 5 4 3 2 1
READY

\[ \begin{align*}
&
\end{align*} \]
Analysis of **FOR-NEXT STEP** Statements

<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>KEY WORD</th>
<th>COUNTER VARIABLE</th>
<th>INITIAL VALUE</th>
<th>FINAL VALUE</th>
<th>INCREMENT/DECREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>FOR</td>
<td>J</td>
<td>10</td>
<td>1</td>
<td>STEP -1</td>
</tr>
<tr>
<td>20</td>
<td>PRINT</td>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>NEXT</td>
<td>J</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The FOR-NEXT STEP loop works as follows: The first time the FOR statement is executed, the counter is set for the initial value "10." Then it executes Line 20 (PRINT J). When the program reaches Line 30 (NEXT J), the counter is decremented by the amount specified (Step-1). If this step has a positive value, the counter is incremented by the amount specified (e.g., Step 2 means increment by 2's).
Comparison of \textbf{GOTO}, \textbf{IF-THEN}, and \textbf{FOR-NEXT} Program Loops

A. \textbf{GOTO} (Unconditional Loop)

\begin{verbatim}
5 PRINT CHR$(125)
10 PRINT "AUBREY"
20 GOTO 10
RUN
\end{verbatim}

- Program loops one zillion times! (or until you stop it)

B. \textbf{IF-THEN} (Conditional Loop)

\begin{verbatim}
5 PRINT CHR$(125)
10 LET J = 0
20 J = J + 1
30 IF J>6 THEN 99
40 PRINT "AUBREY "; J
50 GOTO 20
99 END
RUN
\end{verbatim}

- This program loops 6 times!

C. \textbf{FOR-NEXT} (Conditional Loop)

\begin{verbatim}
5 PRINT CHR$(125)
10 FOR J = 1 TO 6
20 PRINT = "AUBREY "; J
30 NEXT J
99 END
RUN
\end{verbatim}

- This program loops 6 times!
Comparison of **GOTO**, **IF-THEN**, and **FOR-NEXT** Program Loops

A. **"DUMB LOOP"**
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- AUBREY
- READY

B. **"SMART LOOP"**
- AUBREY 1
- AUBREY 2
- AUBREY 3
- AUBREY 4
- AUBREY 5
- AUBREY 6
- READY

C. **"SMART LOOP"**
- AUBREY 1
- AUBREY 2
- AUBREY 3
- AUBREY 4
- AUBREY 5
- AUBREY 6
- READY

**NOTE:** Press the **BREAK** key to get out of loop.
FOR-NEXT Summary

- **FOR—NEXT** is always used as a pair.
- If the key word "step" is not used, the increment of 1 is assumed.
- If the step has a negative value, the counter is decremented (e.g., for \( J = 10 \) to 1 step -1).
- If the step has a positive value, the counter is incremented (e.g., for \( J = 4 \) to 10 step 2).
Flowchart Symbols

- Begin or End
- Processing Block
- Decision Diamond
- Connector Arrows
GOTO-LOOP
(Unconditional)

(LOOP)
20
GOTO 10

10
PRINT "AUBREY"
Looping with **IF-THEN**

**FUNCTION**

- **Clears Screen**
  - PRINT CHR$(125)

- **Initialized Program**
  - 10 LET J = 0

- **Counter**
  - 20 J = J + 1

- **Decision Block**
  - IS J > 6?
    - YES: 30 PRINT "AUBREY "; J
    - NO: (LOOP)

  - (LOOP) GOTO 20

**END**
Looping with **FOR-NEXT**

1. PRINT CHR$(125)
2. 10 FOR* = 1 TO 6
3. (LOOP)
4. NEXT* J
5. 30 IS J > 6
6. NO 20 PRINT "AUBREY"; J
7. YES 99 END

*FOR-NEXT* *Work together as a counter*
Timer Loop

• The Atari can do approximately 500 FOR-NEXT loops per second.

• Example
  5 REM 15 SECOND TIMER PROGRAM
  10 PRINT “TIMER PROGRAM COUNTING”
  20 FOR X = 1 TO 7500
  30 NEXT X
  40 PRINT “TIMER PROGRAM ENDED”

• You don’t believe the Atari can count?
  Well, try it! (Type in the above program and RUN.)
  — Don’t forget to use your watch!
Assignment 10-1

1. Type, enter, and RUN the following program.

   5 PRINT "INPUT A VALUE N": PRINT: PRINT
   10 PRINT "ENTER 1500, 2500, 3500, or 7500"
   15 INPUT N
   20 PRINT CHR$(125)
   25 PRINT "THIS IS A DEMONSTRATION OF"
   30 PRINT: PRINT
   35 FOR J = 1 TO N: NEXT J
   40 PRINT "USING A FOR-NEXT TIMER LOOP"
   45 PRINT: PRINT: PRINT: PRINT
   50 FOR J = 1 TO N: NEXT J
   60 PRINT "TO CHANGE THE DISPLAY'S SPEED"
   65 PRINT: PRINT
   70 FOR J = 1 TO N: NEXT J
   80 PRINT "CHANGE THE VALUES OF N IN THE FOR-NEXT LOOP"
   85 PRINT: PRINT: PRINT: PRINT
   90 FOR J = 1 TO N: NEXT J
  100 PRINT "IF YOU WISH TO STOP THIS DISPLAY"
  105 PRINT: PRINT
  110 FOR J = 1 TO N: NEXT J
  120 PRINT "PRESS THE BREAK KEY"
  130 FOR J = 1 TO N: NEXT J
  140 GOTO 20

2. Make certain that you understand this program and can explain it to your teacher.
PRACTICE 18

Counting Programs Using IF-THEN and FOR-NEXT

1. Using IF-THEN, write a program to count 5's from 50 to 5.
   a. Written vertically
   b. Written horizontally
2. Do not type NEW (that is, save the program above).
3. Using FOR-NEXT, write a program to count to 50 by 5's written horizontally.
   Note: Start your second program at Line 100. That is, type Line 100 as follows: 100 PRINT:
   PRINT (Of course, this is to insert two spaces between your outputs.)
4. How many program lines (excluding Line 100) did it take using FOR-NEXT? ______
   How many using IF-THEN? ______
5. What can you conclude from this task?

PRACTICE 19

Using IF-THEN and FOR-NEXT Statements

1. Using IF-THEN, write a program to generate all the even numbers between 11 and 51 from
   smallest to the largest (that is, 12, 14, 16, and so forth).
2. Do not type NEW.
3. Using FOR-NEXT, write a program that generates the same numbers and PRINT them
   horizontally. (Note: Start at Line 100. Type Line 100 as → 100 PRINT: PRINT and your next
   line should be 110.)
4. Type NEW and enter.
5. Using IF-THEN, write a program to generate all even numbers between 11 and 51 from
   largest to the smallest.
6. Do the same using FOR-NEXT.
PART 11

Reading Data

What You Will Learn

1. To explain the purpose and use of the key words READ, DATA, RESTORE.
2. To compare the three different ways you have learned to input data into the Atari.
3. To write, enter, and run programs using READ-DATA and READ-RESTORE key words.
READ-DATA

READ-DATA statements are much more efficient than INPUT or LET statements when you have lots of data to input.
Ways of Inputting Data to the Computer
(i.e., Ways We've Learned So Far)

10 LET A = 5
10 INPUT A
10 DATA 5
20 READ A

BUILT-IN  FROM KEYBOARD  READ-DATA COMBINATION
# Ways of Inputting Data to the Computer

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 LET A = 5</td>
<td>• LET statement builds value into the program.</td>
</tr>
<tr>
<td>OR</td>
<td></td>
</tr>
<tr>
<td>10 INPUT A</td>
<td>• INPUT statement allows you to enter data through the keyboard.</td>
</tr>
<tr>
<td>OR</td>
<td></td>
</tr>
<tr>
<td>10 DATA(5)</td>
<td>• DATA statement contains the value (5), which will be stored in a specified variable.</td>
</tr>
<tr>
<td>20 READ A</td>
<td>• READ statement names the variables in which the values are to be stored.</td>
</tr>
</tbody>
</table>

NOTES: Data lines can be read only by READ statements. The READ-DATA work together to input data to the computer.
READ-DATA Example

5 REM READ-DATA EXAMPLE

DATA STATEMENT → 10 DATA 1, 2, 3, 4, 5

READ STATEMENT → 20 READ A, B, C, D, E

PRINT STATEMENT → 30 PRINT A, B, C, D, E

NOTES:
- Each piece of data must be read by a READ statement.
- Each READ statement can read a number of pieces of data if each variable is separated by a comma.
- Data lines can only be used by READ statements.
Exercise 11-1 (Reading Data)

Type and enter.

10 DATA 1,2,3,4,5
20 READ A,B,C,D,E
30 PRINT A, B, C, D, E

Type RUN and press [RETURN].

1 2 3 4
5

NOTES:
• The display shows that all five pieces of data in Line 10 were read by Line 20, assigned letters A through E, and printed by Line 30.
• Data lines are always read left to right by READ statements.
READ-DATA Summary (Key Words)

DATA
- Key word that lets you store data inside your program to be accessed (read) by READ statements.
- Data items will be read sequentially starting with the first item in the first DATA statement and ending with the last item in the last DATA statement.
- Items in data list may be string or numeric constants.
- When using string values in DATA statements, all characters will appear as seen in the Data line (e.g., quotes, spaces, colons, etc.).
- Commas can NEVER appear in a string DATA statement except to separate DATA statements.
- DATA statements must match up with the variable types in the corresponding READ statement.
- DATA statements may appear anywhere it is convenient in a program.

• EXAMPLE:
  10 DATA JONES A.B., SMITH R.J.
  20 DATA LEEDS MIDDLE SCHOOL, COMPUTERS
  30 DATA 125, 250, 750, 1000
READ-DATA Summary (Key Words)

READ

• Key word that instructs the computer to read a value from a DATA statement and assign that value to the specified variable.
  — The first time a READ statement is executed, the first value in the first DATA statement is used; the second time, the second value in the DATA statement is used. When all the items in the first DATA statement are used (READ), the next READ will use the first value in the second DATA statement, and so on.
  — An out-of-data error occurs if there are more attempts to READ than there are data items.

• EXAMPLE:

40 DIM A$(10), B$(10), C$(20), D$(10)
50 READ A$, B$, C$, D$, A, B, C, D

(Note that there are eight READ variables and eight DATA items on previous page for program Lines 10, 20, and 30)
Assignment 11.1

1. Type and enter the following program:

5 DIM A$(20)
10 PRINT "NAME", "GRADE"
20 READ A$
30 IF A$ = "END" THEN PRINT "END OF LIST": END
40 READ G
50 IF G < 75 THEN PRINT A$, G
60 GOTO 20
70 DATA GRAY BILL, 95, JONES A.B., 65
80 DATA JONES A.C., 100, SMITH R.L., 70
90 DATA EPPS S.W., 60, WELLS DAVE, 100, END

2. Predict the output of the program.

3. Why was the DIM statement used?

4. RUN the program and record the results.
RESTORE

• Key word that causes the next READ statement executed to start over with the first DATA statement.
  — This lets your program reuse the same data lines.
  — Sometimes it is necessary to READ the same data more than once without having to run the complete program again; therefore, RESTORE is used.
  — Whenever the program comes to RESTORE, all data lines are restored to their original unread condition, both those lines that have been READ and those that have not been READ. This allows all data to be available for reading again, starting with the first data item in the first data line.

NOTE! Remember that each piece of data in a data line can only be read once each time the program is RUN. The next time a READ statement requests a piece of data, it will READ the next piece of data in the data line, or, if data on that line are all used up, it will go to the next data line and start reading it. Therefore, the RESTORE statement is needed if the same data is to be READ more than once in the same program.
Illustration of the READ-RESTORE Feature

```
10 DATA 1, 2, 3, 4, 5
20 ...   FOR N = 1 TO 5
30 READ A
35 PRINT A; " ";
40 RESTORE
50 NEXT N
RUN
1 1 1 1 1
```

NOTE:
- RESTORE caused data Line 10 to be restored to its original unread condition, making all data available for reading again.
- Since there is only one read variable, A, it starts with the first piece of data, 1, in this case.
Exercise 11-2 (READ-RESTORE Data in a FOR-NEXT Loop)

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
</table>
| 1. Type and enter.             | 10 DATA 1,2,3,4,5  
|                                | 20 FOR N = 1 TO 5  
|                                | 30 READ A  
|                                | 40 PRINT A ;;  
|                                | 50 NEXT N |
| 2. Type RUN and press RETURN   | 1 2 3 4 5  
|                                | READY    |
| 3. Insert Line 35. (Type and enter) | 35 RESTORE | Restores Data Line to Its Original Unread Condition |
| 4. Type RUN and press RETURN   | 1 1 1 1 1  
|                                | READY    | Therefore Computer Reads First Data Item Over and Over |
READ-DATA Summary

- **READ-DATA**
  - Key words used to input lots of data to the computer.
- **RESTORE**
  - Key word used to restore (put back) data so it can be used again.
- Data lines can be read only by READ statements.
  - If more than one piece of data is placed on a data line, they must be separated by commas.
    Each piece of data must be read by a READ statement.
- Data lines are read from left to right by READ statements.
  - Data lines can be placed anywhere in a program.
- **READ-DATA** statements are extremely common.
  - RESTORE is used less often.
PRACTICE 20

READ-DATA

1. Type and enter the following program:
   
   ```
   5 PRINT CHR$(125)
   10 PRINT "NAME", "GRADE"
   15 DIM A$(20)
   20 READ A$
   30 IF A$ = "END" THEN PRINT "END OF LIST":END
   40 READ G
   50 IF G > 75 PRINT A$, G
   60 GOTO 20
   70 DATA GRAY BILL, 95, JONES A.B., 65
   80 DATA JONES A.C., 100, SMITH R.L., 70
   90 DATA EPPS S.W., 60, WELLS DAVE, 100, END
   ```

2. Predict the output of the program.

3. RUN the program and record the results.
What You Will Learn

1. To explain the purpose of key words COLOR, GRAPHICS, PLOT, POSITION, SOUND, STICK, DRAWTO.

2. To become familiar with the layout of Atari display using the Video Display Worksheets.

3. To draw pictures and letters on the screen.

4. To write and run programs using all the concepts learned in this lesson.

NOTE: The Atari provides the user with an unlimited number of possibilities of graphic application. The student should experiment with graphics. This lesson will introduce the student to some of the basic features of graphics used on the Atari, but we will only "scratch the surface." Students will find out by themselves what other kinds of things can be done with graphics on the Atari.
POSITION X,Y

- Permits you to put the cursor anywhere on the screen.
- X must be from 0 to 39.
- Y must be from 0 to 23.
- Must be used in a program (i.e., you need a line number).
- Uses absolute moves only relative to the borders of the screen (that is, place the cursor anywhere without regard to text or graphics mode).

EXAMPLE

NEW
10 PRINT CHR$(125)
20 FOR K = 0 TO 23
30 POSITION K,K
40 PRINT K
50 NEXT K
In Class Exercise 12-1

1. Type, enter, and RUN the following program.

   5 PRINT CHR$(125)
   10 POSITION 14,0:PRINT "POSITION DEMO"
   20 POSITION 10,10
   30 PRINT "THIS IS AN EXAMPLE"
   40 POSITION 18,15
   50 PRINT "OF"
   60 POSITION 5,20
   70 PRINT "USING POSITION FOR FORMATTING"

2. Run the program several times. Analyze the program and make certain you understand it.

3. Experiment with POSITION (if you have the time).
Atari Graphics

- This lesson assumes a black and white monitor is used. If you are using a color monitor refer to the Atari manual for additional information.
- The Atari has many graphics modes. We will be using GRAPHICS 5. Refer to the Atari BASIC Reference Manual for a discussion of additional graphics modes.
- To get back to the text mode, use the following command: GRAPHICS 0.
- When you use the GRAPHICS 5 command, most of the screen is used for graphics except for four (4) lines at the bottom which are used for text.
- To clear the screen in graphics mode, use the following command: GRAPHICS 0.
Atari
Video Display Layout Showing X, Y, Coordinates

LINE 1 THIS SPACE IS LEFT AT THE BOTTOM
LINE 2 OF THE SCREEN FOR TEXT, YOU
LINE 3 CAN USE THESE LINES TO WRITE YOUR
LINE 4 PROGRAM OR TO PRINT MESSAGES TO A USER
### GRAPHICS COMMANDS

--- GRAPHICS 5, COLOR, PLOT, GRAPHICS 0 ---

<table>
<thead>
<tr>
<th>Key Word</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAPHICS 5</td>
<td>Converts screen to graphics mode</td>
<td>• GRAPHICS 5</td>
</tr>
<tr>
<td>COLOR</td>
<td>Sets the color for plotting in low-resolution graphics mode (there are 4 colors available; they are numbered from 0 to 3)</td>
<td>• COLOR 0 (black)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COLOR 1 (white)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COLOR 2 (green)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COLOR 3 (orange)</td>
</tr>
<tr>
<td>PLOT X, Y</td>
<td>Places a dot at the location specified by the X, Y coordinates. (That is, PLOT allows you to turn on or light up a spot at location X, Y)</td>
<td>• PLOT 0,0 (dot in upper left corner)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PLOT 0,39 (dot in lower left corner)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PLOT 79,0 (dot in upper right corner)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PLOT 79,39 (dot in lower right corner)</td>
</tr>
</tbody>
</table>

- The color of the spot is determined by the most recent value of COLOR, which is 0 (black) if not previously specified
- X value ranges from 0 to 79
- Y value ranges from 0 to 39

- GRAPHICS 0  | Converts entire screen back to text (words) mode. Display can handle up to 40 characters/line and 24 lines. |
In-Class Exercise 12-2 (COLOR, GRAPHICS 5, PLOT)

1. Type:
   GRAPHICS 5
   COLOR 1 (If you don't type this key word, you will not see the graphics
   because the color is set to zero (black) by GRAPHICS command)

2. Locate the following points on your video display worksheet:
   (a) PLOT 9,9   (e) PLOT 20,20   (i) PLOT 36,10
   (b) PLOT 39,0   (f) PLOT 10,20   (j) PLOT 10,36
   (c) PLOT 0,39   (g) PLOT 6,36   (k) PLOT 28,38
   (d) PLOT 39,39   (h) PLOT 8,18   (l) PLOT 38,28

3. Type and enter* the above coordinates in your Atari
   (a) Do they match the points you picked on your worksheet?
   (b) What happens if you plot 95,13? Explain.
   (c) What happens if you plot 20,45? Explain.

4. To clear the screen in graphics mode, type GRAPHICS 0 and press
   RETURN.

*To enter press RETURN (You know this by now!)
Atari
Video Display Worksheet for Graphics 5 Mode
In-Class Exercise 12-3
Drawing Lines on Worksheet

Using the video display worksheet:

1. Plot the following X, Y coordinates on your worksheet:
   - PLOT 10, 20
   - PLOT 11, 20
   - PLOT 12, 20
   - PLOT 13, 20
   - PLOT 14, 20
   - PLOT 15, 20
   - PLOT 16, 20
   - PLOT 17, 20
   - PLOT 18, 20
   - PLOT 19, 20
   - PLOT 20, 20

2. When you finish plotting the above coordinates, you will have a line on your sheet from column (X) ________ to column (X) ________ at row (Y) _________.

3. Plot the following coordinates on the same worksheet used above:
   - PLOT 15, 15
   - PLOT 15, 16
   - PLOT 15, 17
   - PLOT 15, 18
   - PLOT 15, 19
   - PLOT 15, 20
   - PLOT 15, 21
   - PLOT 15, 22
   - PLOT 15, 23
   - PLOT 15, 24
   - PLOT 15, 25
   - PLOT 15, 26

4. When you finish plotting the above coordinates (3), you will have a line on your sheet from row (Y) ________ to row (Y) ________ at column (X) _________.

5. Both plots in (1) and (3) above could be used as an ________ axis for graphs.
In-Class Exercise 12-4
Drawing Lines on Atari (The Hard Way)

1. Horizontal lines
(a) Set COLOR 1 and enter information below (don’t forget to press RETURN after each line).
   COLOR 1
   PLOT 10, 20      PLOT 16, 20
   PLOT 11, 20      PLOT 17, 20
   PLOT 12, 20      PLOT 18, 20
   PLOT 13, 20      PLOT 19, 20
   PLOT 14, 20      PLOT 20, 20
   PLOT 15, 20

(b) What happened? ____________________________

2. Vertical Lines
(a) Set COLOR 1 and enter information below (don’t forget to press RETURN).
   COLOR 1
   PLOT 15, 15      PLOT 15, 16
   PLOT 15, 16      PLOT 15, 17
   PLOT 15, 17      PLOT 15, 18
   PLOT 15, 18      PLOT 15, 19
   PLOT 15, 19      PLOT 15, 20
   PLOT 15, 20

(b) What happened? ____________________________

3. Make up some coordinates on your own and try it!
Summary and Assignment 12-1
GRAPHICS 5, COLOR, PLOT, GRAPHICS Ø

Summary

1. Screen is divided into 40 vertical columns and 80 horizontal rows.
   - X is the horizontal coordinate counting across from the left-hand side of the screen. X coordinate goes from 0 to 79.
   - Y is the vertical coordinate counting from the top of the screen. Y coordinate goes from 0 to 39.

2. PLOT X,Y lights up a spot on the screen.
   - If you try to plot outside the range of X and Y (e.g., PLOT 2, 700 or PLOT -15, -30), you will get a message of either “ERROR-3” or “ERROR-141.”
   - Although the highest number you can use with the Y coordinate is 47, don’t do it! A Y coordinate in the range 40 to 47 will not show up on the screen.

3. Assignment 12-1
   Experiment with PLOT command on your own time until you feel comfortable with it.
In-Class Exercise 12-5
Drawing Lines on Atari (The Easy Way)

1. Horizontal Lines
   (a) Type and enter the following program:
       COLOR 1
       PLOT 0,20: DRAWTO 79,20
       RETURN
   (b) What happened? _______________________
   (c) How many PLOT statements would you need to type to draw the above line the hard way? _______________________

2. Vertical Line
   (a) Type and enter the following program:
       COLOR 1
       PLOT 40,0: DRAWTO 40,39
       RETURN
   (b) What happened? _______________________

3. Try some other examples. Play with DRAWTO until you feel comfortable using this key word to draw lines.

4. Can you think of other things you can draw using the DRAWTO command? (Try some, if you have the time.)
Summary of Graphing Lines

• PLOT A*,B*:DRAWTO C*,D* will draw a line from A*,B* to point C*,D*.

*In an actual example, each of these letters would be replaced with a number between 0 and 39 on Y axis and 0 to 79 on X axis.

10 GRAPHICS 5
20 COLOR 1
30 PLOT 20,1:DRAWTO 24,1
40 PLOT 20,3:DRAWTO 24,3
50 PLOT 20,1:DRAWTO 20,6
60 PLOT 24,1:DRAWTO 24,6
80 PLOT 30,7:DRAWTO 34,7
100 PLOT 32,7:DRAWTO 32,12
130 PLOT 36,14:DRAWTO 40,14
140 PLOT 36,16:DRAWTO 40,16
150 PLOT 36,14:DRAWTO 36,19
160 PLOT 40,14:DRAWTO 40,19
180 PLOT 44,21:DRAWTO 44,26
190 PLOT 44,21:DRAWTO 48,21
200 PLOT 44,23:DRAWTO 48,23
210 PLOT 45,23:DRAWTO 48,26
220 PLOT 48,21:DRAWTO 48,23
230 PLOT 56,28:DRAWTO 56,33
240 PLOT 54,28:DRAWTO 58,28
250 PLOT 54,33:DRAWTO 58,33
260 PRINT CHR$(125)
270 PRINT "VOILA! ATARI GRAPHICS"

• Now RUN the program several times.
**In-Class Exercise 12-6***

Match the letter printed in Column I with the line numbers in Column II.

<table>
<thead>
<tr>
<th>I</th>
<th>II</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A</td>
<td>a. 210-240</td>
</tr>
<tr>
<td>2. T</td>
<td>b. 30-60</td>
</tr>
<tr>
<td>3. R</td>
<td>c. 180-190</td>
</tr>
<tr>
<td>4. I</td>
<td>d. 80-110</td>
</tr>
</tbody>
</table>

5. Change the color value in Line 20 to 0 (COLOR 0). Before running the program, predict what will happen.

6. Write a program to print the first letter of your name.

*Refer to program on page 173.
SUMMARY OF GRAPHICS AND JOYSTICKS

- Joystick controls are very powerful tools in the manipulation of screen information.
- The program below shows how to draw pictures on the screen using a joystick.
- Type and enter:

```plaintext
30  GRAPHICS 7
40  COLOR 1
50  LX=159:LY=95
60  X1=INT(LX/2):Y1=INT(LY/2)
70  IF X1<0 THEN X1=LX
80  IF X1>LX THEN X1=0
90  IF Y1<0 THEN Y1=LY
100 IF Y1>LY THEN Y1=0
110  PLOT X1,Y1
120  FOR Z=1 TO 50:NEXT Z
130  J=STICK(0)
140  IF J=15 THEN GOTO 130
150  IF J>4 AND J<8 THEN GOTO 190
160  IF J>8 AND J<12 THEN GOTO 230
170  IF J=13 THEN Y1=Y1+1:GOTO 70
180  Y1=Y1-1:GOTO 70
190  X1=X1+1
200  IF J=6 THEN Y1=Y1-1:GOTO 70
210  IF J<>7 THEN Y1=Y1+1:GOTO 70
220  GOTO 70
230  X1=X1-1
240  IF J=10 THEN Y1=Y1-1:GOTO 70
250  IF J<>11 THEN Y1=Y1+1:GOTO 70
260  GOTO 70
```

In-Class Exercise 12-7

1. Write the program so that you cannot move off one side of the screen and reappear on the other side.
2. What happens if Line 120 is deleted?
MOVING THE CURSOR

- Another use of graphics and controllers is moving a cursor on the screen. The program below illustrates this point.

- Type and enter:
  10 GRAPHICS 0
  20 LIST
  30 J=STICK(0)
  40 FOR Z=1 TO 10:NEXT Z
  50 REM LINE 40 SLOWS CURSOR MOVEMENT
  60 IF J=15 THEN GOTO 30
  70 IF J>12 THEN GOTO 180
  80 IF J>8 AND J<12 THEN GOTO 140
  90 REM MOVE CURSOR WITH ARROW CHARACTERS
  100 PRINT CHR$(31);
  110 IF J=6 THEN PRINT CHR$(28);
  120 IF J=5 THEN PRINT CHR$(29);
  130 GOTO 30
  140 PRINT CHR$(30);
  150 IF J=10 THEN PRINT CHR$(28);
  160 IF J=9 THEN PRINT CHR$(29);
  170 GOTO 30
  180 IF J=14 THEN PRINT CHR$(28);
  190 IF J=13 THEN PRINT CHR$(29);
  200 GOTO 30

- Hit the [BREAK] key when finished.

In-Class Exercise 12-8

1. What happens if Line 40 is deleted?
2. Write the program so the cursor cannot move “off” the screen.
SUMMARY OF ATARI GRAPHICS CAPABILITY

• The Atari computer system is a capable graphics computer. The three programs below will illustrate these capabilities. Notice how figures can be magnified and reduced using different graphics modes.
• Type and enter:

```
20 GRAPHICS 5
30 COLOR 1
40 PLOT 26,14:DRAWTO 40,14
50 DRAWTO 48,6:DRAWTO 36,6
60 DRAWTO 26,14:DRAWTO 26,30
70 DRAWTO 40,30:DRAWTO 48,22
80 DRAWTO 36,22:DRAWTO 26,30
90 PLOT 36,6:DRAWTO 36,22
100 PLOT 48,6:DRAWTO 48,22
110 PLOT 40,14:DRAWTO 40,30
120 FOR I=1 TO 10000:NEXT I
130 END
```

• Run this program several times.

In-Class Exercise 12-9
1. Write the program using GRAPHICS 7. What happened to the cube?
3. Center the cube by changing the corners of the cube.
4. Enlarge the cube to twice its original size.
ATARI SOUND CAPABILITY

- The following three programs will give insight into using the Atari sound system. The last program is a jet passing overhead. The other three programs are note-generation programs.

- Type and enter:

  10 SOUND 0,121,10,8  
  20 SOUND 1,96,10,8  
  30 SOUND 2,81,10,8  
  40 SOUND 3,60,10,8  
  50 FOR I=1 TO 2000:NEXT I  
  60 SOUND 0,0,0,0  
  70 SOUND 1,0,0,0  
  80 SOUND 2,0,0,0  
  90 SOUND 3,0,0,0  
 100 END

In-Class Exercise 12-10

1. What happens if Line 50 is deleted?
2. Retype Line 50 and use different values. What happens?

- Type and enter:

  10 FOR J=1 TO 43  
  20 READ A,B  
  30 FOR I=1 TO A  
  40 SOUND 0,B,10,10  
  50 NEXT I  
  60 NEXT J  
  70 DATA 50,96,50,108,50,121,50,108  
  80 DATA 50,96,5,0,50,96,5,0,75,96  
  90 DATA 10,0,50,108,5,0,50,108  
 100 DATA 5,0,75,108,10,0,50,96  
 110 DATA 5,0,50,81,5,0,75,81,5,0,50,108  
 120 DATA 50,96,50,108,50,121,50,108  
 130 DATA 50,96,5,0,50,108,5,0,50,96  
 140 DATA 5,0,50,96,5,0,50,108,5,0  
 150 DATA 50,108,5,0,50,96,5,0,50,108  
 160 DATA 5,0,100,121  
 170 END

3. Describe what happens when Line 30 is changed to FOR I=1 TO
   (a) A/2
   (b) A/3
   (c) A/10
   (d) 2*A
   (e) 3*A
   • Run the program several times.
In-Class Exercise 12-11

• Type and enter:
  10 FOR I=1 TO 100
  20 SOUND 0,9,8,I
  30 NEXT I
  40 FOR A=10 TO 140
  50 SOUND 0,10,8,A/10
  60 NEXT A
  70 FOR B=10 TO 50
  80 SOUND 0,B,8,15
  90 NEXT B
 100 FOR C=140 TO 1 STEP-1
 110 SOUND 0,50,8C/10
 120 NEXT C
 130 END

1. Change the divisor values in Lines 50 and 110 and see (or hear) what happens.
SOME FUN PROGRAMS TO TRY ON YOUR OWN

- The three programs below are short (but sweet) sound or graphics demonstrations for the Atari. Have fun!

- Type and enter:

```plaintext
5 REM PHASOR FIRE SOUND PROGRAM
10 FOR I=0 TO 120
20 SOUND 0,1,8,10
30 FOR X=1 TO 5
40 NEXT X
50 NEXT I
60 END
```

- Type and enter:

```plaintext
10 REM CANNON GRAPHICS PROGRAM
20 GRAPHICS 7
30 COLOR 1
40 PLOT 10,79
50 DRAWTO 15,70
60 PLOT 10,75
70 DRAWTO 15,79
80 PLOT 10,75
90 DRAWTO 10,79
100 FOR X=-7 TO 7 STEP 0.2
110 Y=X^2
120 COLOR 1
130 PLOT X*10+85,Y+20
140 FOR I=1 TO 10
150 NEXT I
160 COLOR 0
170 PLOT X*10+85,Y+20
180 NEXT X
190 PRINT "BOOM!"
200 END
```

- Type and enter:

```plaintext
5 REM NOISE GENERATION SOUND PROGRAM
10 FOR I=0 TO 31
20 SOUND 0,1,12,10
30 FOR X=1 TO 25
40 NEXT X
50 NEXT I
60 FOR I=1 TO 200
70 NEXT I
80 END
```
PRACTICE 21

Graphics

1. Write a program that will do the following:
   a. Draw a horizontal line across the top of the screen (Line 0).
   b. Add the necessary steps to your program to draw a vertical line down the middle of the screen.
   c. Add the necessary steps to your program to draw a horizontal line across the bottom of the screen (last line of the display).
   d. Add the necessary steps to draw a vertical line on the far left side of the display.
   e. Add the necessary steps to draw a vertical line on the far right side of the display.
   f. Enter and RUN your program.

   Display should look like this after part (E).
PART 13

Arrays

What You Will Learn

1. To explain the purpose of using arrays.
2. To set up one- and two-dimensional numeric arrays.
3. To explain the purpose and use of the terms DIM, A(3), A(2,3), DIM A(10), DIM DB(7,5).
4. To develop, enter, and run programs using numeric arrays.
Arrays

A. What is an array?
   • An array is a lineup, an arrangement, or an orderly grouping of things.

B. Why use an array?
   • Use it when we wish to have more variables available in a program.
     — Although the Atari BASIC permits the use of approximately 128 variables for numerics, sometimes thousands of variables are required for storing and retrieving many pieces of data.
     — The array allows you to arrange your data so that it can be stored and retrieved easily.

C. How to use a numeric array in Atari BASIC*
   • The Atari must be instructed how much memory space to reserve for an array.
     — This is done by using the DIM statement.
     — To use the arrays A and B on the next page, you would write:
       DIM A(6), B(6).
     — DIM stands for DIMension. When you DIMension an array, you tell the Atari the maximum number of array elements your array can have.
     — DIM A(6) actually reserves 7 spaces for the array, since A(0) is the first element of the array.
     — Computers, unlike people, begin counting with the number 0.

*Other computers use arrays differently, so when using another computer, be careful!
One-Dimensional Array — Illustration

SIX-ELEMENT ARRAY — NAMED A*

- A(3) is pronounced A SUB 3.
  - A(3) represents the third cell or box in the array (lineup).
  - Data stored in this cell would be addressed by the label A(3).
  - Suppose data were stored in the sixth cell: A(6)? (You got it!)

SIX-ELEMENT ARRAY — NAMED B*

- B(5) represents the fifth cell in the array where data can be stored and retrieved.

*A and B are optional names. Any valid variable name can be used to name an array in Atari BASIC.
One-Dimensional Array — Program Example

PROGRAM

5 DIM A(6)
10 DATA 100, 200, 300, 400, 500, 600
20 FOR W = 1 to 6
30 READ A(W)
40 NEXT W

50 FOR W = 1 to 6
60 PRINT W, A(W)
70 NEXT W
RUN

DISPLAY

1 100
2 200
3 300
4 400
5 500
6 600

REMARKS

• Lines 20-40 store data in array A(W)
• Lines 50-70 retrieve data from array A(W)
One-Dimensional Array — Program Example (Con’t)

ARRAY
CONTENTS

A(W)
A(1) → 100
A(2) → 200
A(3) → 300
A(4) → 400
A(5) → 500
A(6) → 600

Above is an illustration of what happens after data are stored in array A(W). Note that in location A(1), the first data element (100) is stored. In location A(2), the second data element (200) is stored, and so on until the sixth data element (600) is stored in location A(6). Remember that Line 10 of the program contained the data elements that were read using Lines 20 through 40.
Two-Dimensional Array — Illustration

<table>
<thead>
<tr>
<th>ROW</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
</tr>
</tbody>
</table>

36 ELEMENT ARRAY (MATRIX) (NAMED H)

H(3,4) REFERS TO CELL OR BOX ON ROW 3, COLUMN 4
In-Class Exercise 13.1

(Fill in the Blanks Using the Matrix)

<table>
<thead>
<tr>
<th>LABEL</th>
<th>ROW</th>
<th>COLUMN</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>H(1,1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(4,5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(3,3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(2,3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(6,6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(1,6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(2,4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(4,4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DIM Statement

- Review the DIM statement on page 183.

- EXAMPLE
  
  ```
  10 DIM A(6), B(2,3), C(21)
  ```

  Sets a one-dimension array A with 6 elements
  A(0) — A(5)
  or
  A(1) — A(6)*

  Sets a one-dimension array with 21 elements
  A(0) — A(20) or A(1) — A(21)*

  Sets a two-dimension array B
  with 3 ROWS (numbered 0-2)
  and 4 COLUMNS (numbered 0-3)

*If A(0) is not used
Checkbook Array Example

- Consider the following table of checkbook information:

<table>
<thead>
<tr>
<th>Check #</th>
<th>Date Written</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6/5/81</td>
<td>$15.50</td>
</tr>
<tr>
<td>101</td>
<td>6/7/81</td>
<td>25.00</td>
</tr>
<tr>
<td>102</td>
<td>6/15/81</td>
<td>145.00</td>
</tr>
<tr>
<td>103</td>
<td>6/22/81</td>
<td>65.00</td>
</tr>
<tr>
<td>104</td>
<td>6/30/81</td>
<td>211.00</td>
</tr>
<tr>
<td>105</td>
<td>6/30/81</td>
<td>79.50</td>
</tr>
</tbody>
</table>

- Note that every item in the table may be specified by reference to two numbers: the row number and the column number. For example, (Row 3, Column 3) refers to the amount $145.00.

- The above table can be set up in a $6 \times 3$ array or matrix (see next page).
Checkbook Array Example (Con’t)

<table>
<thead>
<tr>
<th>CK</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>60581</td>
<td>15.50</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>60781</td>
<td>25.00</td>
</tr>
<tr>
<td>3</td>
<td>102</td>
<td>61581</td>
<td>145.00</td>
</tr>
<tr>
<td>4</td>
<td>103</td>
<td>62281</td>
<td>65.00</td>
</tr>
<tr>
<td>5</td>
<td>104</td>
<td>63081</td>
<td>211.00</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
<td>63081</td>
<td>79.50</td>
</tr>
</tbody>
</table>

6 x 3 MATRIX (ARRAY) — NAMED CK

NOTES:

1. Data recorded in form mm ddyy where mm = month number, dd = day, and yy = last two digits of year.

2. Since CK is a numeric array, alpha-numerical characters such as dashes cannot be stored.
Checkbook Array Example (Con’t)

YOUR ACTION

1. Setting Up the Array
   (Lines 10 through 110)
   A. Let's type and enter Lines 10 through 110 as shown:

   (NOTE: Line 10 sets up dimension of array. Lines 20-150 read the values into array CK.)
   NOTE: DIM CK (6, 3) Sets up a 6 × 3 array (excluding zero subscripts) with 6 rows (numbered 1 to 6) and 3 columns (numbered 1 to 3)

2. Manipulating the Array
   (Finding the Sum)
   A. Add Lines 160 through 190 to the program as shown:

   (NOTE: Lines 120-160 add up all the checks written.)

   B. Type RUN and press RETURN.

   DISPLAY

   10 DIM CK(6,3)
   20 FOR ROW = 1 TO 6
   30 FOR COL = 1 TO 3
   40 READ K
   50 CK(ROW,COL) = K
   60 NEXT COL
   70 NEXT ROW
   100 DATA 100,60581,15.50
   110 DATA 101,60781,25.00
   120 DATA 102,61581,145.00
   130 DATA 103,62281,65.00
   140 DATA 104,63081,211.00
   150 DATA 105,63081,79.50
   160 FOR ROW = 1 TO 6
   170 SUM = SUM + CK(ROW,3)
   180 NEXT ROW
   190 PRINT "TOTAL AM’T OF CKS $ “; SUM

   TOTAL AM’T OF CKS $541
Checkbook Array Example (Con’t)

YOUR ACTION

3. Manipulating the Array
(Print out all checks written on a given day)

A. Do not type NEW.

B. Add the following steps to your program:

C. Type RUN and press RETURN.

D. Enter a date (e.g., 63081 which is 6/30/81).

DISPLAY

```
195 PRINT:PRINT
200 PRINT "LIST CKS FROM (MM DD YY)";
205 INPUT DT: PRINT:PRINT
210 PRINT "CKS WRITTEN ON ",DT;
220 PRINT " ARE LISTED BELOW"
225 PRINT:PRINT
230 PRINT "CHECK ", "AMOUNT".PRINT
240 FOR ROW = 1 TO 6
250 IF CK(ROW,2) = DT THEN PRINT CK(RO
W,1), CK(ROW,3)
260 NEXT ROW

TOTAL OF CHECKS WRITTEN $541.00
LIST CHECKS WRITTEN ON (MM DDYY)?
CHECKS WRITTEN ON 63081 ARE LISTED BELOW;
CHECK # AMOUNT
104 211
105 79.5
```
Assignment 13-1

Summary

• A2 ≠ A(2)
  — A2 is an ordinary variable
  — A(2) is a subscripted variable

• Any time you use an array, you must use a DIM statement.
  — Example:
    10 DIM A(25), B(17, 18)

• One-Dimensional Array
  SUBSCRIPT
  — A(3) is pronounced A SUB 3
  NAME

• Two-Dimensional Array (Matrix)
  ROW
  — H(3,4) refers to cell or box on row 3, column 4
  NAME COLUMN
PRACTICE 22

Arrays

1. Write a program to read the following numbers into an array and then PRINT them out:
   676  150  175  188  190  277  876  976  912  544
2. Change program to find the sum and average of the 10 numbers given.
3. Label the answer: The sum is ________, and the average is ________.

PRACTICE 23

One-Dimensional Array

1. Suppose we had the following results of a quiz given to a class of 10 students:

<table>
<thead>
<tr>
<th>Student #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student's Grade</td>
<td>75</td>
<td>85</td>
<td>95</td>
<td>87</td>
<td>100</td>
<td>77</td>
<td>83</td>
<td>69</td>
<td>98</td>
<td>88</td>
</tr>
</tbody>
</table>

   a. Using a one-dimensional Array, write a program to find the class average.
   b. Add the necessary program lines to find the highest grade and the lowest grade.
   c. Have the program PRINT : Class Average is ________, Highest Grade is ________, and Lowest Grade is ________.
   d. Enter and RUN each of these programs several times.
PART 14

INT(X), ABS(X) & RND(X) Functions

What You Will Learn

1. To explain the purpose and use of INT(X), ABS(X), and RND(X) functions.
2. To write, run, and analyze programs using the INT(X), ABS(X), and RND(X) functions.
INT(X) Function

• INT(X) or integer function allows you to round off any number, large or small, positive or negative, into a whole number (or integer).

• INT(X) means
  — If X is a positive number, then the largest whole number can be found by chopping off the decimal part.
  
  Example:
  \[
  \text{INT}(5.7) = 5 \\
  \text{INT}(0.7) = 0 \\
  \]

  — If X is a negative number, the largest whole number can be found by moving down to the next lowest whole number (that is, make a negative number more negative).

  Examples:
  \[
  \text{INT}(-0.6) = -1 \\
  \text{INT}(-0.2) = -1 \\
  \text{INT}(-3.14) = -4 \\
  \text{INT}(-7.28) = -8 \\
  \]
**Exercise 14-1 INT(X)**

**Graphical Representation**

For negative numbers: "Move to next lowest whole number"

For positive numbers: "Chop off decimal part"

<table>
<thead>
<tr>
<th>X</th>
<th>INT(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>-1.7</td>
<td></td>
</tr>
<tr>
<td>2.345</td>
<td></td>
</tr>
<tr>
<td>-0.8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3.1415</td>
<td></td>
</tr>
<tr>
<td>76.14</td>
<td></td>
</tr>
<tr>
<td>-10.35</td>
<td></td>
</tr>
</tbody>
</table>
INT(X) FUNCTION — ROUNдинG $$

1. Type and enter this program.
2. Now RUN.
3. Add Line 15 to program as shown. (Note: In Line 15 we multiply by 100, add .5, take the INT, which is now 667, and then divide 667 by 100. 667/100 is 6.67, which is what we want, two decimal places.)
4. Now RUN program.
Assignment 14-1 INT(X)

1. Type NEW and enter this program for finding the area of a circle:

   10 REM AREA OF A CIRCLE 3.14159* R □ 2
   20 PRINT "THE RADIUS IS";
   25 INPUT R
   30 P = 3.14159
   40 A = P*R □ 2
   50 PRINT "THE AREA IS "; A

2. RUN the program several times to make sure it works.

3. Change the program to suppress (chop off) all of the numbers to the right of the decimal point. (RUN the program to make sure it works.)

4. Change the program to make the answer accurate to one decimal place. (For example, if R = 1, then Area (A) = 3.1.)
ABS(X) Function

- \( \text{ABS}(X) = \) Abbreviation for absolute value of \( X \)
- Examples:
  
  \[
  \begin{align*}
  \text{ABS} (12) &= 12 \\
  \text{ABS} (0) &= 0 \\
  \text{ABS} (-10) &= 10 \\
  \text{ABS} (-357) &= 357
  \end{align*}
  \]
- Note! \( \text{ABS} (25 - 10) = \text{ABS} (10 - 25) = 15 \)

Assignment 14-2 ABS(X)

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Type and enter the program shown.</td>
<td>10 PRINT &quot;TYPE ANY POS. OR NEG. &quot;;</td>
</tr>
<tr>
<td></td>
<td>15 INPUT N</td>
</tr>
<tr>
<td></td>
<td>20 X = ABS (N)</td>
</tr>
<tr>
<td></td>
<td>30 PRINT &quot;N&quot;, &quot;X&quot;</td>
</tr>
<tr>
<td></td>
<td>40 PRINT N, X</td>
</tr>
<tr>
<td>2. RUN the program several times using both positive and negative numbers.</td>
<td>(Note that regardless of the number you input as ( N ), the absolute value of ( X ) is the same number without the sign.)</td>
</tr>
</tbody>
</table>
RND(X) Function

• RND(X) or random number function causes the computer to give you a “surprise” number.
  - It’s as though the computer spins a wheel of chance.
  - It’s like pulling a number out of a hat.
  - It’s unpredictable!

• The random number function – general form
  Let \( N = \text{INT} (X \times \text{RND}(1)) + 1 \)
  Where \( N = \) The random number
  \( \text{RND} = \) Abbreviation for random
  \( X = \) Any number between 1 and 32767

• The general form for finding random numbers may seem a little complicated at first but it’s not once you understand how to use it. All you need to do is just give “X” the value or number you wish to be the highest random number. When you run the program, you will have a number between 1 and X.
Example:
10 PRINT INT (4 * RND (1)) +1 (will give you a random number from 1 to 4 inclusive)
20 PRINT INT (6 * RND (1)) +1 (will give you a random number from 1 to 6 inclusive)
30 PRINT INT (10 * RND (1)) +1 (will give you a random number from 1 to 10 inclusive)

• Type, enter, and RUN the above program several times or until you understand how random numbers work.
Random Number — Program Example

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Type and enter. (Line 5 allows you to enter “X” or the highest random number you want.)</td>
<td>5 PRINT “ENTER A # FROM 1 TO 100”; 10 INPUT X 20 FOR J = 1 TO 10 30 PRINT INT (X * RND (1)) + 1 40 NEXT J  (SCREEN SHOULD HAVE TEN RANDOM NUMBERS BETWEEN 1 AND X.)</td>
</tr>
<tr>
<td>2. Type RUN and press <strong>RETURN</strong>.</td>
<td></td>
</tr>
<tr>
<td>3. RUN program again to get the idea.</td>
<td>10 FOR J = 1 TO 100  (SCREEN SHOULD HAVE ONE HUNDRED RANDOM NUMBERS BETWEEN 1 AND X.)</td>
</tr>
<tr>
<td>4. Change Line 10 to read:</td>
<td></td>
</tr>
<tr>
<td>5. RUN. (Get the idea?)</td>
<td></td>
</tr>
</tbody>
</table>
Coin Toss Program

ACTION AND REMARKS

1. Type and enter program as shown:

   (Line 20 initializes counters, sets \( H = T = 0 \).)
   (Line 40 starts next line at top of screen.)
   (Line 60 begins FOR-NEXT statement and runs it “N” times.)
   (Line 70 generates integers between 1 and 2.)
   (Line 80 tells the program to go to Line 90 if \( X = 1 \) = heads and to Line 100 if \( X = 2 \) = tails.)
   (Line 90, “heads” are counted.)
   (Line 100, “tails” are counted.)
   (Line 110 sends control back to Line 60 for “N” passes.)

DISPLAY

10 REM H = HEADS, T = TAILS
20 H = 0: T = 0: PRINT
30 PRINT “HOW MANY FLIPS OF THE COIN”;
35 INPUT N
40 PRINT CHR$(125)
50 PRINT “COIN IS FLIPPING... STANDBY”
60 FOR K = 1 TO N
70 X = INT (2 * RND (1)) + 1
80 ON X GOTO 90, 100
90 H = H + 1 : GOTO 110
100 T = T + 1
110 NEXT K
120 PRINT CHR$(125)
# Coin Toss Program (Con’t)

<table>
<thead>
<tr>
<th>ACTION AND REMARKS</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Line 130 prints the headings.)</td>
<td>130 PRINT &quot;HEADS&quot;, &quot;TAILS&quot;, &quot;TOTFLIPS&quot; : PRINT</td>
</tr>
<tr>
<td>(Line 140 prints the values of H, T, and N.)</td>
<td>140 PRINT H, T, N</td>
</tr>
<tr>
<td>(Line 150 calculates and prints the percentage of heads, percentage of tails.)</td>
<td>150 PRINT 100<em>H/N; &quot;%&quot;, 100</em>T/N; &quot;%&quot;</td>
</tr>
<tr>
<td>(Line 160 provides spacing for better appearance.)</td>
<td>160 PRINT: PRINT: PRINT</td>
</tr>
</tbody>
</table>
Assignment 14-3 RND(X)

YOUR ACTION

1. Type and enter the program as shown.

2. RUN the program.

DISPLAY

5 REM PICK A NUMBER GAME
10 PRINT CHR$(125)
20 X = INT (10 * RND (1)) +1
30 PRINT "ENTER A # FROM 1 TO 10";
35 INPUT N
40 IF X = N THEN 100
50 IF X < N THEN 110
60 IF X > N THEN 120
100 PRINT "RIGHT ON": GOTO 10
105 FOR J = 1 TO 2500: NEXT: GOTO 10
110 PRINT "LOWER": GOTO 30
120 PRINT "HIGHER": GOTO 30
Assignment 14-3 RND(X)

3. Analyze the program.
   Line 10 ____________ the display.
   Line 20 is the ____________ generator.
   Line 30 allows the user to ____________ a number.
   Lines 40, 50, and 60 are ________________ statements that compare
   conditional, unconditional
   the random number ___________ with the input number ___________.
   \[ X, N \]
   \[ X, N \]
   Lines 100, 110, and 120 are PRINT statements that guide the player.
   Why does Line 105 GOTO Line 10 and why do Lines 110 and 120 GOTO
   Line 30? Explain the function of Line 105.

4. Modify (change) the program to pick a number between 1 and 100, and RUN
   this program several times.
Summary

- **ABS(X)** — Provides the absolute value of X regardless of the number you input (i.e., X is that same number without the sign).

- **INT(X)** — Provides integer or whole number value of X.
  - If X is a positive (+) number, it chops off the decimal part.
  - If X is a negative number, it rounds down to the next lowest whole number (e.g., INT (−0.6) = −1).

- **RND(X)** — Causes the computer to give you a random number.
  - INT (X * RND (1)) + 1 gives you a random number from 1 to X inclusive.
PRACTICE 24

INT(X) and ABS(X)

1. Fill in the banks with the appropriate INT(X):

<table>
<thead>
<tr>
<th>X</th>
<th>INT(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>-2.5</td>
<td></td>
</tr>
<tr>
<td>6.365</td>
<td></td>
</tr>
<tr>
<td>-0.8</td>
<td></td>
</tr>
<tr>
<td>-10.65</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3.2425</td>
<td></td>
</tr>
<tr>
<td>-7.61</td>
<td></td>
</tr>
<tr>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>

2. The following program can be used for finding the area of a circle:

```
10 REM AREA OF A CIRCLE = 3.14159*RA^2
15 DIM A$(10)
20 PRINT "THE RADIUS IS";
23 INPUT R
25 PRINT "RADIUS IN (IN.,FT,OR YD,)";
27 INPUT A$
30 A = 3.14159*RA^2
40 PRINT "THE AREA IS "; A; " SQ. "; A$
```

a. Enter and RUN the program several times to make certain it works.
b. Change the program to suppress (chop off) all the numbers to the right of the decimal point (RUN the program to make sure it works).
c. Change the program to make the answer accurate to one decimal place. (For example if \( R = 1 \), then area \( A = 3.1 \)).

PRACTICE 25

Random Number

1. Write a program that will let you pick a random number between 1 and 100. The program should let you input a number from the keyboard and provide the following clues on your guess.
   a. If the number you pick matches the number the computer picks, have the computer PRINT "Right On."
   b. If the number from the keyboard is too high, have the program print "Lower."
   c. If the number from the keyboard is too low, have the program print "Higher."
   d. Enter and RUN the program several times.
PART 15

Subroutines

What You Will Learn

1. To explain the purpose for using subroutines.

2. To explain the purpose and use of terms ON-GOTO, GOSUB, RETURN, ON-GOSUB.

3. To develop, enter, and run programs using subroutines and ON-GOTO statements.
Subroutine

What Is It?
• A subroutine is a short program or routine that is built into a large program to do specific calculations or perform repetitive functions.

Why Use It?
• There are times when you need the same type of calculation at various points in your program, but instead of retyping the statements needed for this calculation each time, you can write a subroutine to perform the needed calculations.

How Do You Call a Subroutine?
• To call or branch to a subroutine, use the GOSUB statement.
  — The GOSUB XXXXX statement directs the computer to go to that line number and execute the program steps until it reaches the key word RETURN, which ends the subroutine.
  — RETURN is always built into a subroutine and is used to tell the computer that the subroutine is finished. When finished, the control of the program is returned to the statement in the main program immediately following the most recently executed GOSUB.
Subroutine Example

Main Program:

10 REM GOSUB EXAMPLE
20 
90
100 GOSUB 3000
110 PRINT "BACK FROM SUBROUTINE": END

Subroutine:

3000 PRINT "EXECUTING THE SUBROUTINE"
3010
3040
3050 RETURN
Subroutine Illustration

Main Program

10 REM MAIN PROGRAM BEGINS HERE
   ...
100 GOSUB 1000
110 REM MAIN PROGRAM CONTINUES
   ...
200 GOSUB 2000
210 REM MAIN PROGRAM CONTINUES
   ...
290 END REM MAIN PROGRAM ENDS

Subroutines

1000 REM SUBROUTINE #1
   ...
1060 RETURN

2000 REM SUBROUTINE #2
   ...
2050 RETURN
Subroutine Illustration (Con't)

1. When the computer reaches the GOSUB in Line 100, the program will branch (GOTO) Line 1000, which is the beginning of Subroutine #1.

2. After Subroutine #1 is executed and the RETURN (Line 1060) is reached, control is passed back to the main program (Line 110). Note that Line 110 is the next higher number after the GOSUB that put it in the subroutine (Line 100).

3. The computer continues through the main program to the GOSUB in Line 200, which branches control to Subroutine #2 in Line 2000.

4. After the subroutine is executed, the RETURN (Line 2050) passes the control back to Line 210 in the main program. (Note again that this is the next higher line number after the GOSUB in Line 200.)

5. An END statement is included in the program (Line 290) after the main program is finished to keep it from accidentally falling into the subroutine. We only want the subroutines to be executed when we call for them by a GOSUB.
Sample Program Using Subroutines
(Temperature Conversion)

Main Program

10 REM TEMPERATURE CONVERSION PROGRAM
15 PRINT CHR$(125): DIM A$(1)
20 PRINT "CONVERT C TO F (Y OR N)";: INPUT A$
30 IF A$ = "Y" THEN 80
35 PRINT
40 PRINT "INPUT DEGREES FAHRENHEIT";: INPUT F
50 GOSUB 2000
60 PRINT "ARE YOU FINISHED (Y OR N)";: INPUT A$
70 IF A$ = "N" THEN 40
75 END
80 PRINT "INPUT DEGREES CENTIGRADE";: INPUT C
90 GOSUB 1000
100 PRINT "ARE YOU FINISHED (Y OR N)";: INPUT A$
110 IF A$ = "N" THEN 80
120 END

Subroutine #1

1000 REM CELSIUS TO FAHRENHEIT CONVERSION
1010 F = (9/5) * C + 32 : PRINT
1020 PRINT C; " DEG. CELSIUS ="; F; " DEG. FAHRENHEIT"
1030 RETURN

Subroutine #2

2000 REM FAHRENHEIT TO CELSIUS CONVERSION
2010 C = (F-32) * (5/9): PRINT
2020 PRINT F; " DEG. FAHRENHEIT ="; C; " DEG. CELSIUS"
2030 RETURN
Analysis of Sample Program Using Subroutines

1. Lines 10 through 110 comprise the main program.
2. Line 20 is an input statement to ask the user if he wants to convert from C to F or from F to C. Yes (Y) means C to F and No (N) means F to C.
3. Line 30 is a conditional branch statement. If the user wants to convert Centigrade C to Fahrenheit, then branch to Line 80; otherwise, skip a line (PRINT) and go to Line 40.
4. Line 40 allows the user to input the °F to be converted to °C.
5. Lines 50 and 90 call the subroutines.
6. Line 60 asks the user if he is finished. In Line 70 the program will branch to Line 40 (if B$ = N) or the program will END (if B$ ≠ N).
7. Line 80 is similar to Line 40, except that it allows the user to input the °C to be converted to °F.
8. Lines 100 and 110 are the same as Lines 60 and 70.
9. The first subroutine begins at Line 1000 and ENDS at Line 1030. It RETURNS control to Line 100 in the main program.
10. The second subroutine begins at Line 2000 and ENDS at Line 2030. It RETURNS control to Line 60 in the main program.
Subroutine Exercise

10 PRINT "THIS IS" ; " ";
20 GOSUB 1000
30 PRINT "OF HOW" ; " ";
40 GOSUB 2000
50 PRINT "WORKS"
60 END
1000 PRINT "AN EXAMPLE" ; " ";
1010 RETURN
2000 PRINT "A SUBROUTINE" ; " ";
2010 RETURN

1. Analyze the program and write the message. ____________
2. Now type and enter the program.
3. RUN the program. Does it agree with your message?
Assignment 15-1

1. Analyze the program below and write the message:

10 LET B = 10
20 GOSUB 2000
30 B = B + 5
40 GOSUB 2000
50 B = B + 10
60 GOSUB 2000
99 END

2000 REM SUBROUTINE
2010 IF B<12 THEN 2050
2020 IF B = 25 THEN 2070
2030 PRINT "PRIME"
2040 GOTO 2080
2050 PRINT CHR$(125): PRINT "LEEDS"
2060 GOTO 2080
2070 PRINT "COMPUTERS"
2080 RETURN

Message ____________________________________________
## ON-GOTO Example

### YOUR ACTION

1. Type NEW and enter this program.

2. Before you RUN the program, analyze it. Can you predict what will happen when you RUN it? (I sure hope you can by now!)

3. RUN the program several times until you feel comfortable with it.

### DISPLAY

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>PRINT CHR$(125)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>PRINT &quot;TYPE A NUMBER FROM 1 TO 3&quot;;</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>INPUT N</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>IF N = 1 THEN 110</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>IF N = 2 THEN 130</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>IF N = 3 THEN 150</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>PRINT &quot;REMEMBER, # IS FROM 1 TO 3&quot;</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>GOTO 10</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>PRINT &quot;N = 1&quot;</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>PRINT &quot;N = 2&quot;</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>PRINT &quot;N = 3&quot;</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
ON-GOTO Example (Con’t)

<table>
<thead>
<tr>
<th>YOUR ACTION</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Erase Lines 20, 30, and 40. (Remember, there are two ways to do this! Simply type in each line number separately and then press \texttt{RETURN}).</td>
<td>20 ON N GOTO 110, 130, 150</td>
</tr>
<tr>
<td>5. Type and enter this line:</td>
<td>(SHOULD HAVE NEW LINE 20 \ PLUS LINES 5, 10, AND 50 THROUGH 160 FROM PREVIOUS PAGE. IF YOU DON'T HAVE THESE LINES, FIX IT!)</td>
</tr>
<tr>
<td>6. List your program.</td>
<td>(WORKS JUST THE SAME AS BEFORE, DOESN'T IT?)</td>
</tr>
<tr>
<td>7. RUN the program a few times.</td>
<td>N = 1</td>
</tr>
<tr>
<td>8. RUN the program again.</td>
<td>N = 1</td>
</tr>
<tr>
<td>Use the following inputs:</td>
<td>N = 2</td>
</tr>
<tr>
<td>1.5</td>
<td>\texttt{N = 2}</td>
</tr>
<tr>
<td>1.8</td>
<td>\texttt{REMEMBER, # IS FROM 1 TO 3!}</td>
</tr>
<tr>
<td>2.8</td>
<td>N = 3</td>
</tr>
<tr>
<td>0.8</td>
<td>READY</td>
</tr>
<tr>
<td>3.99</td>
<td>■</td>
</tr>
<tr>
<td>(Now do you understand that N = \texttt{INT (N)} or whole number?)</td>
<td></td>
</tr>
</tbody>
</table>
ON-GOTO Example Analysis

1. Line 20 tells the computer to do the following:
   • If, the integer (whole number) value of N is 1, GOTO Line 110.
   • If the integer value of N is 2, GOTO Line 120.
   • If the integer value of N is 3, GOTO Line 130.
   • If the integer value of N is not one of the numbers listed above, then move on to the next line.

2. The ON-GOTO statement has a built-in INT statement, which really acts like this:
   20 ON INT (N) GOTO----ETC.
Assignment 15-2 ON-GOTO

1. Type and enter the following program:
   
   5 PRINT CHR$(125)
   10 INPUT "ENTER # FROM 1 TO 5";
   15 INPUT N
   20 ON N GOTO 100, 200, 300, 400, 500
   30 PRINT "# SHOULD BE FROM 1 TO 5!" : GOTO 10
   40 END
   100 PRINT "N = 1" : END
   200 PRINT "N = 2" : END
   300 PRINT "N = 3" : END
   400 PRINT "N = 4" : END
   500 PRINT "N = 5" : END

2. Answer the following questions before running the program:
   a. What happens (output) if the input is 1.8 (Line 10)?
   b. What happens (output) if the input is 3.99?
   c. What happens (output) if the input is 2.89?
   d. What happens if the input is 0.5?

3. RUN the program several times and record the following:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ON-GOSUB

• Works like ON-GOTO, except control branches to one of the subroutines specified by the line numbers in the line number list.

• Example:
  10 PRINT "CHOOSE 1, 2, OR 3";
  15 INPUT K
  20 ON K GOSUB 1000, 2000, 3000
  99 END
  1000 PRINT "SUBROUTINE #1" : RETURN
  2000 PRINT "SUBROUTINE #2" : RETURN
  3000 PRINT "SUBROUTINE #3" : RETURN

• K may be a numerical constant, variable, or expression.
  — It must have a positive value, however, or an error will occur.

• If K ≠ 1, 2, or 3, the program will go to the next line (99 END).
Summary

- **GOSUB XXXX**, causes the computer to:
  - Go to the subroutine beginning at line XXXX (the specified line number).
  - Work through the subroutine until it finds a RETURN statement.
  - Return control to the statement that follows the GOSUB statement in the main program.

- **ON n GOSUB XXXX, ----, YYY**
  - Multi-way branching statement that is controlled by a test variable (n), which sends control of the program to one of the subroutines specified by line numbers in the line number list (i.e., XXXX, ----, YYY).
  - The test variable n must be a numerical constant, variable, or expression that has a non-negative value or else an error will occur.

- **ON n GOTO XXXX, ----, YYY**
  - Works like ON n GOSUB except control branches to one of the line numbers specified (XXXT, ----, YYYY).
  - ON n GOTO 1st line number, 2nd line number ---- nth line number expression must be between 0 and 255 inclusive.
  - If n<0, an error will occur.
PRACTICE 26
Program to Convert Centigrade to Fahrenheit and Vice Versa

1. Write a program that will do the following:
   a. Convert Centigrade to Fahrenheit.
   b. Convert Fahrenheit to Centigrade.
   c. Allow you to select either A or B above.
   d. Allow you to input from keyboard.
   e. PRINT the answer as follows:
      • ___ degrees Celsius = ___ degrees Fahrenheit
      or
      ___ degrees Fahrenheit = ___ degrees Celsius
   * Keyboard input value
   ** Calculated output value

PRACTICE 27
Program for Sample Profit/Loss Statement

1. When a product is sold for more than it costs, the seller receives a profit. When a product is sold for less than it costs, the seller takes a loss.

   Therefore: sell price − cost = profit or loss

   If we let: S = Sell price
              C = Cost
              U = No. of units
              P = Profit
              L = Loss

   Then: P (or L) = S*U − C*U

   a. Write a program that will compute the profit or loss for a business if the sell price and cost are known. (Note: Program should permit you to enter cost and sell price from the keyboard.)
   b. Have the computer PRINT the following:
      
      | NO. OF UNITS |
      | UNIT PRICE ($) |
      | UNIT COST ($) |
      | TOTAL SALES ($) |
      | TOTAL COST ($) |
      | PROFIT/LOSS ($) |
      | % OF SALES |

   c. RUN the program several times and record your answer.
EXTRA PRACTICE 1

Programming Mathematical Operators

1. Given two numbers A=25 and B=5:
   a. Write one program that will add, subtract, divide (A/B), multiply, and square the two numbers (A and B).
   b. The answer should PRINT as shown here:
      The sum of A and B is ______ (your answer).
      The difference of A and B is ______ (your answer).
      The quotient of (A/B) is ______ (your answer).
      The product of A*B is ______ (your answer).
      The square of A is ______ (your answer).
      The square of B is ______ (your answer).

EXTRA PRACTICE 2

Finding the Average

1. Write a program to find the average of three numbers.
2. Have the program PRINT: The average is ______
3. Add a program line to have the program PRINT the average of your #______, your #______, and your #______ is your answer ______. Example: The average of 3, 4, and 8 is 5.

EXTRA PRACTICE 3

More Mathematical Operations

Write five separate programs to PRINT the answer to these problems (the answer should read 25 * 2 + 4 = 54, and so on.):
1. 25*2+4
2. 3² + 4 - 2
3. 36 ÷ 4 *5
4. 28 + 4 * 6 ÷ 8
5. (18−2) ÷ 3 + 4 (6*3) + 2³

EXTRA PRACTICE 4

Print Zones

Part I.

Write a program to PRINT the word “Leeds” in the following ways:

ZONE 1  ZONE 2  ZONE 3
1. LEEDS  LEEDS  LEEDS
2. LEEDS  LEEDS  LEEDS
3.       LEEDS  LEEDS
4.       LEEDS  LEEDS
5.       LEEDS

Part II.

Using page 74:
1. Count the number of characters in all four zones. How many?
2. How many in zone 1 ___________, zone 3, __________.
EXTRA PRACTICE 5

Area of Square and Volume of Cube

1. Write a program to solve the following problems. Label your answers.
   a. The side of a square is 27 inches. Find its area \((A) = s^2\).
   b. If the side of a cube is also 27 inches, find its volume \((V) = s^3\).
2. Using INPUT statements, write a program to find the area of a square and volume of a cube.
   a. Solve the problems above (assume sides of square and cube are equal).
   b. Using different lengths for the side, RUN the program again (assume that the sides of the square and the cube are equal).

EXTRA PRACTICE 6

Printing Tables of Numbers, Squares, and Cubes

1. Write a program to generate the first 25 numbers and PRINT their squares on the same line.
   Example:
   
   \[
   \begin{array}{ccc}
   1 & 1 \\
   2 & 4 \\
   3 & 9 \\
   4 & 16 \\
   \end{array}
   \]
   and so forth

2. Write a program to generate the first 25 numbers and PRINT their cubes on the same line.
   Example:
   
   \[
   \begin{array}{ccc}
   1 & 1 \\
   2 & 8 \\
   3 & 27 \\
   4 & 64 \\
   \end{array}
   \]
   and so forth

3. Write a program to generate all the numbers from 20 to 1 and PRINT the numbers, and their squares and cubes, on the same line and in four columns.
   Example:
   
   \[
   \begin{array}{cccc}
   20 & 400 & 8000 & 160000 \\
   19 & 361 & 6859 & 130321 \\
   18 & 324 & 5832 & 104976 \\
   \end{array}
   \]
   and so forth

EXTRA PRACTICE 7

Printing Three Times and Nine Times Tables

1. Write a program to generate the three times table from \(3 \times 1 = 3\) to \(3 \times 12 = 36\). The printout should look exactly like this:
   
   \[
   \begin{array}{c}
   3 \times 1 = 3 \\
   3 \times 2 = 6 \\
   3 \times 3 = 9 \\
   3 \times 4 = 12 \\
   \end{array}
   \]
   and so forth

2. Write a program to generate the nine times table from \(9 \times 1 = 9\) to \(9 \times 12 = 108\).
# EXTRA PRACTICE 8

## Two-Dimensional Array

1. Suppose we have a class of ten students. The course grade is based upon three quizzes, and the results for the class are as follows:

<table>
<thead>
<tr>
<th>Student #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiz #</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>88</td>
<td>41</td>
<td>100</td>
<td>88</td>
<td>79</td>
<td>76</td>
<td>86</td>
<td>90</td>
<td>85</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>75</td>
<td>52</td>
<td>65</td>
<td>57</td>
<td>98</td>
<td>86</td>
<td>96</td>
<td>91</td>
<td>86</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>47</td>
<td>75</td>
<td>77</td>
<td>86</td>
<td>96</td>
<td>85</td>
<td>92</td>
<td>97</td>
<td>82</td>
</tr>
</tbody>
</table>

a. Write a program to PRINT the following information:

<table>
<thead>
<tr>
<th>Student #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiz #</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and so forth

<table>
<thead>
<tr>
<th>Quiz #</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Computer calculates and PRINTS average