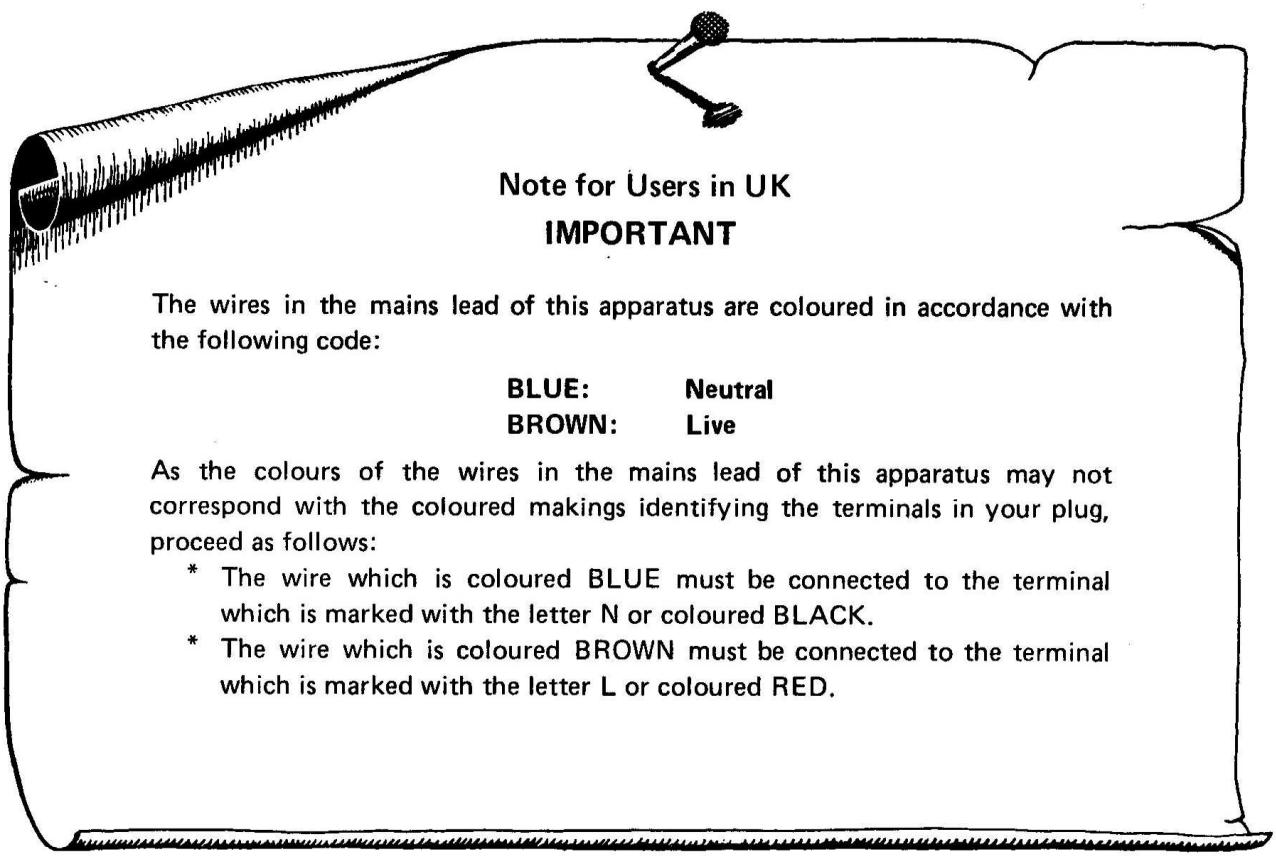
BASIC

MZ-80

SHARP CORPORATION



Note for Users in UK

IMPORTANT

The wires in the mains lead of this apparatus are coloured in accordance with the following code:

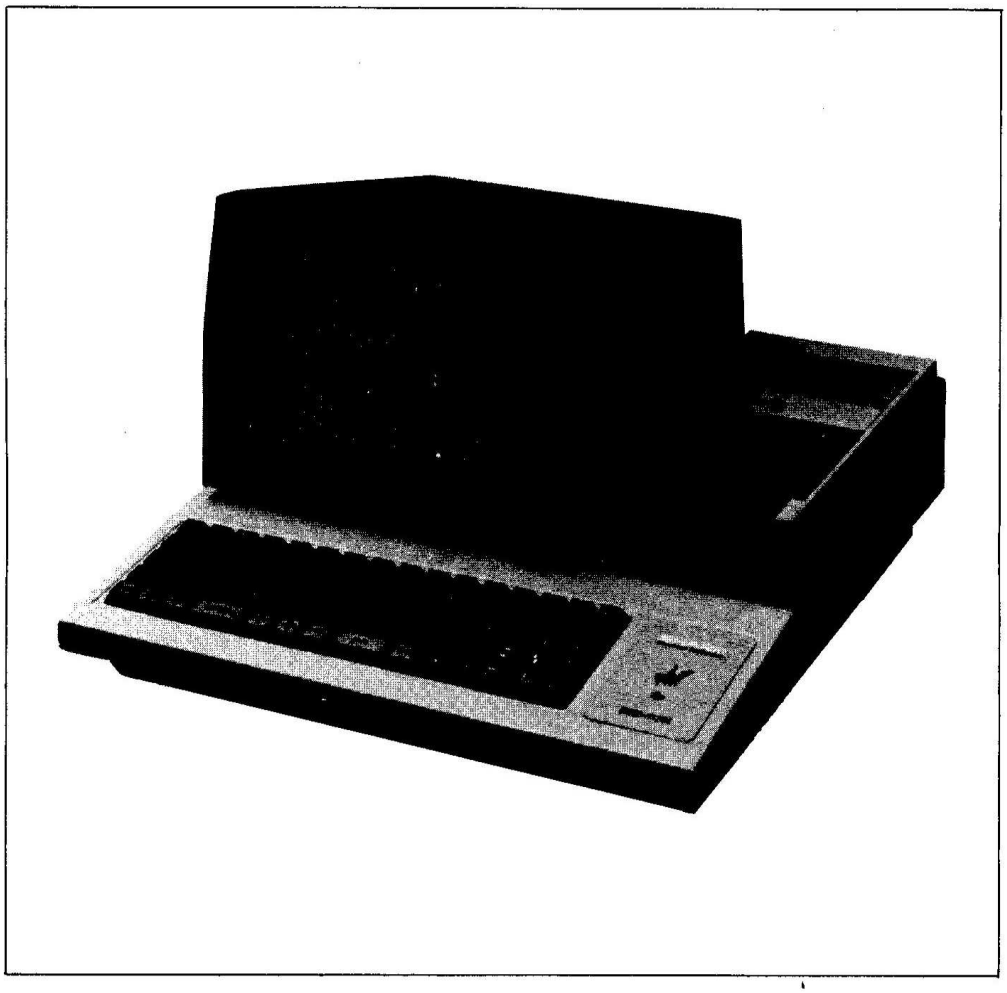
BLUE: **Neutral**
BROWN: **Live**

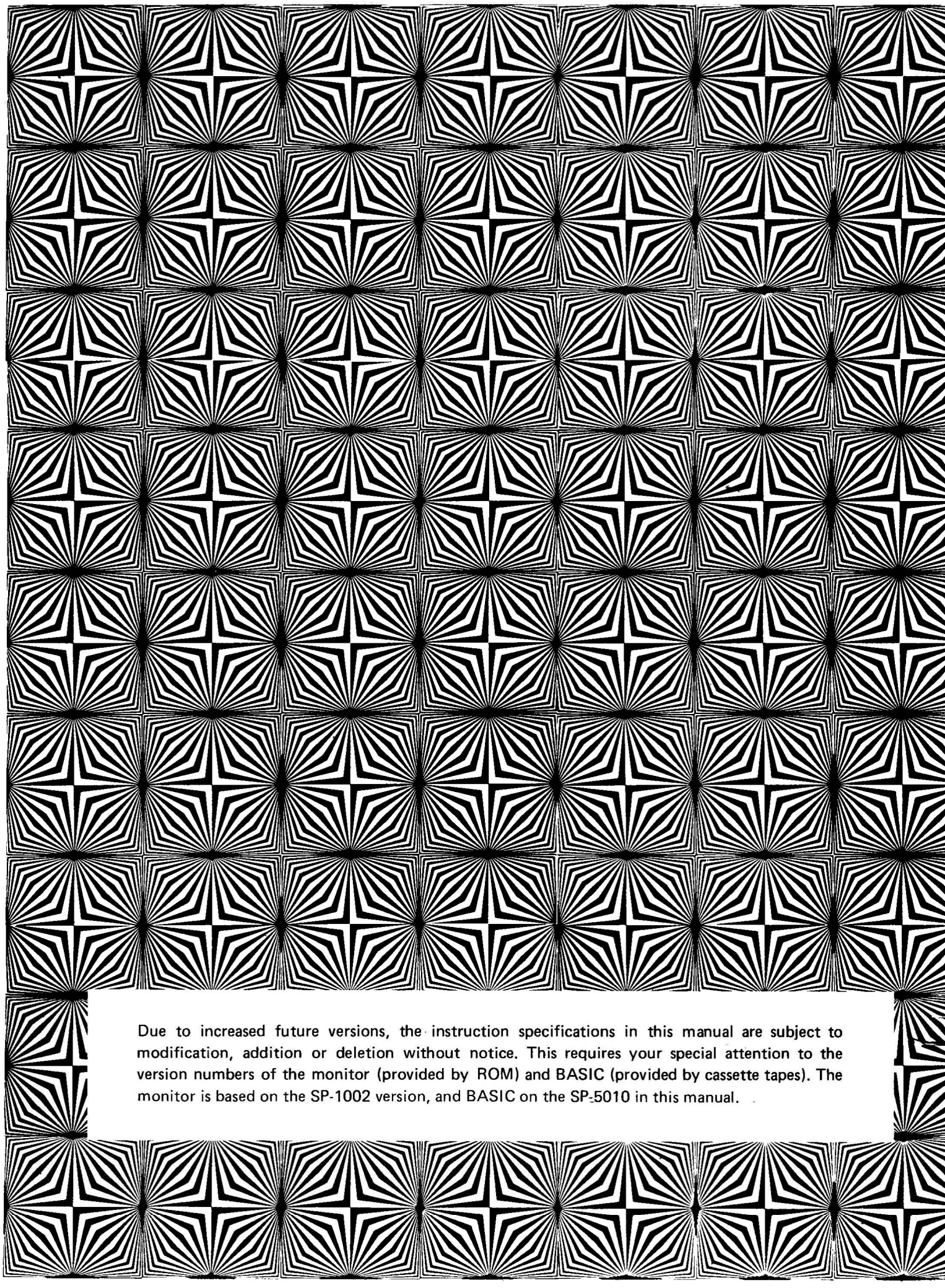
As the colours of the wires in the mains lead of this apparatus may not correspond with the coloured makings identifying the terminals in your plug, proceed as follows:

* The wire which is coloured BLUE must be connected to the terminal which is marked with the letter N or coloured BLACK.
* The wire which is coloured BROWN must be connected to the terminal which is marked with the letter L or coloured RED.

Foreword

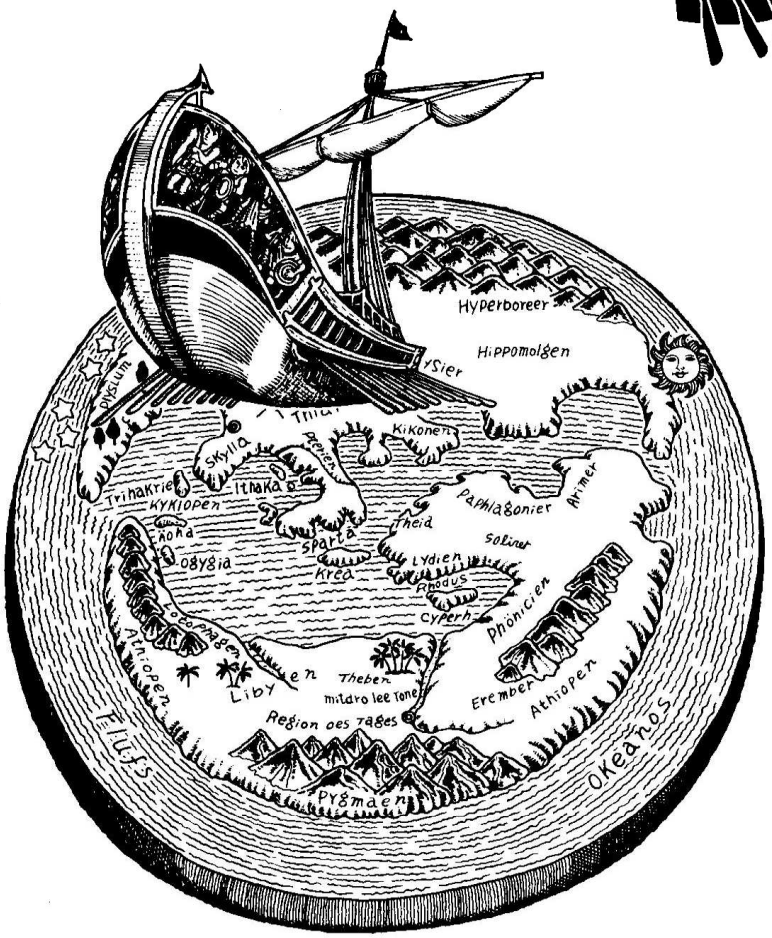
Congratulations on your purchase of a Sharp Personal Computer, the MZ-80K.
An important suggestion for you is to keep this MZ-80K Basic Manual. It will be of great use to tell you what to do if anything goes wrong or you are uncertain of your operation.





Due to increased future versions, the instruction specifications in this manual are subject to modification, addition or deletion without notice. This requires your special attention to the version numbers of the monitor (provided by ROM) and BASIC (provided by cassette tapes). The monitor is based on the SP-1002 version, and BASIC on the SP-5010 in this manual.

MZ-80K
BASIC Manual



From Ancient Times People Have Been Fascinated by the Lights from Outer Space

That's why through times people would have looked at the starry sky while lying down at night. So also would have the lad in Greek mythology dreamt of lights spreading over the vistas of space.
Then, he sailed for the Golden Fleece, aiming at a bright future. Indeed, Argo Navis is such a constellation clearly shining over the horizon in the south. This constellation is symbolized by the ship sailing with youth; the ship representing courage, future, search and longing.
Just like the ship of youth, the MZ-80K will sail before long. To realize your dreams in such a manner, we have created this Sharp's original mark.

Contents

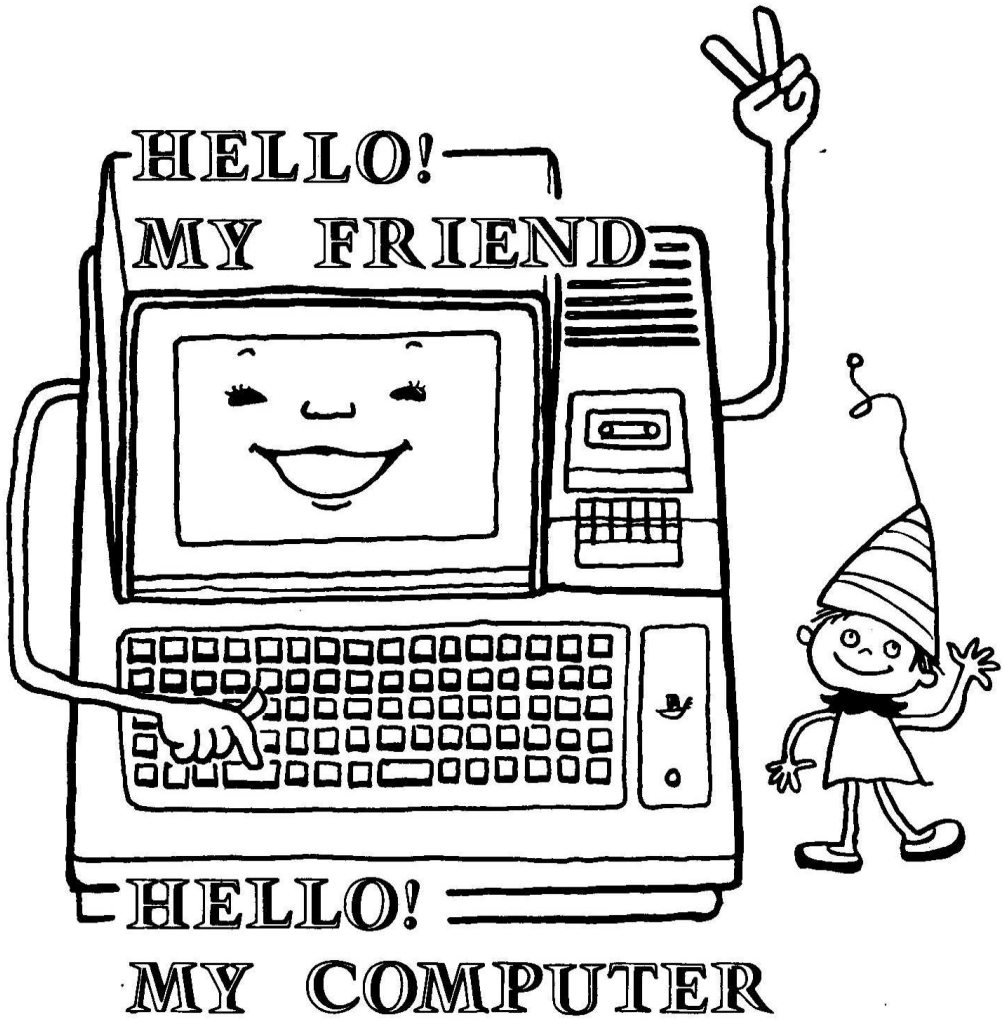
Trying gains Experience.

A "try" is a must to start something new. In fact, your first "try" at anything requires a lot of courage, and even so, it is difficult. You will drown, for example, if you jump into the sea right after reading a book on swimming having trained only at home. This is because you have no "experience," though you know the fact. To gain your "experience," the basic study is essential to have a good "try."

Your computer is not a dangerous tool that puts you at risk if you make a small mistake. It will become too convenient for you to be without once you get used to it.

This manual is designed to help you understand the essence of your computer by learning in your time. With the computer in front of you, use this manual to guide you as you go along.

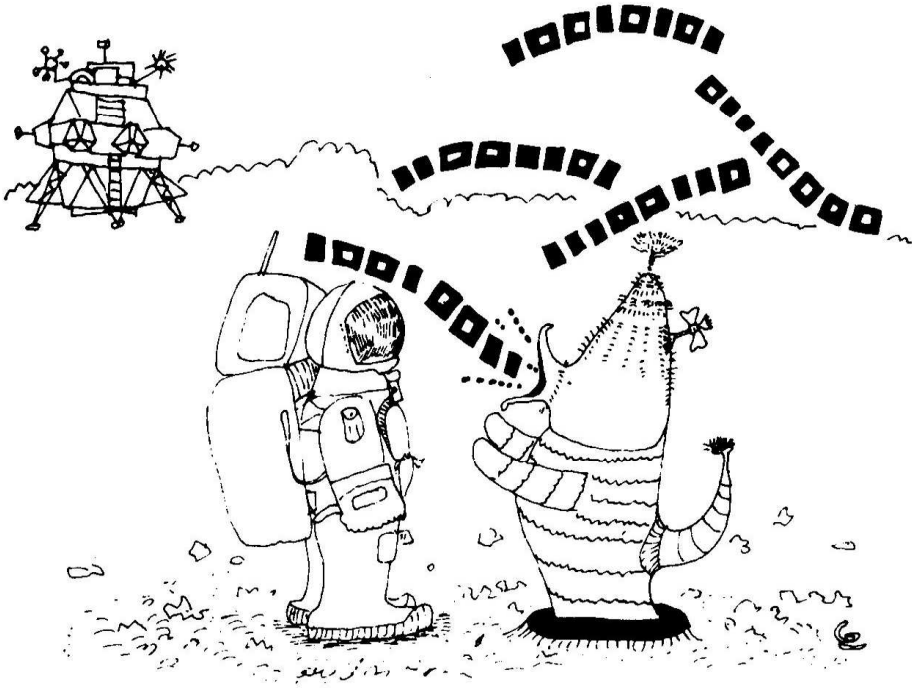
Now, Let's Shake Hands with Me



Here's a new friend for you.

The MZ-80K is ready to enjoy conversation with you. Through conversation, it will help you solve difficult calculation problems or become a partner to play a game with. More than that, it has unknown potentialities to be opened up with you. This is just like a journey into unknown space. Together with your new friend, let's make the journey now.

Welcome to the Land of BASIC!

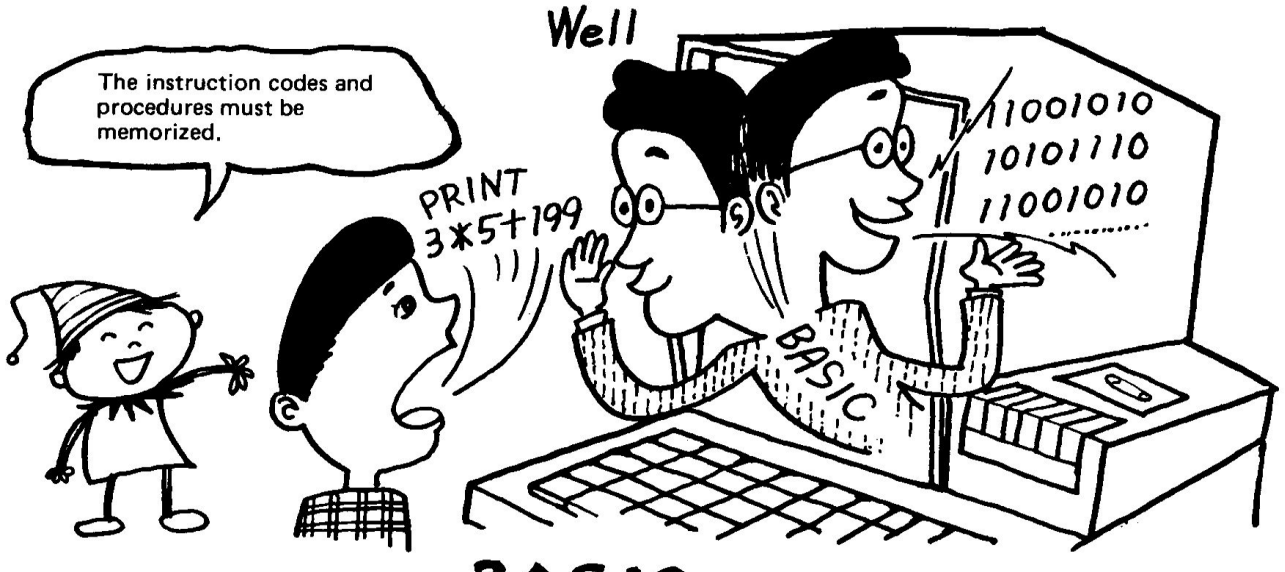


Conversation requires a language. Even those with great ability will not be able to work together without any common language to understand each other. Similarly, a common language is crucial for understanding between a computer and human beings. Using such a common language, instructions can be given to a computer for its execution of a wide variety of jobs that are possible with the combined use of instructions. The common language is termed "Computer Language". Human languages are various in type, such as English, Japanese and German, for example. So are the computer languages according to their applications. The computer language called "BASIC" is widely used for microcomputers.

BASIC is made up of easy English statements featuring the capability of generating a string of instructions for a computer to do the types of jobs, namely, a "program" in the form of conversation between a man and a computer.

Of course, the type of computer language in use is at your disposal.

For the time being, however, description proceeds based on the use of BASIC.

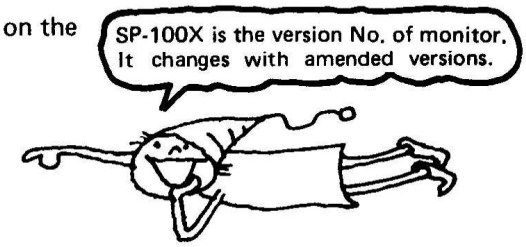


BASIC

Beginner's All-purpose Symbolic Instruction Code

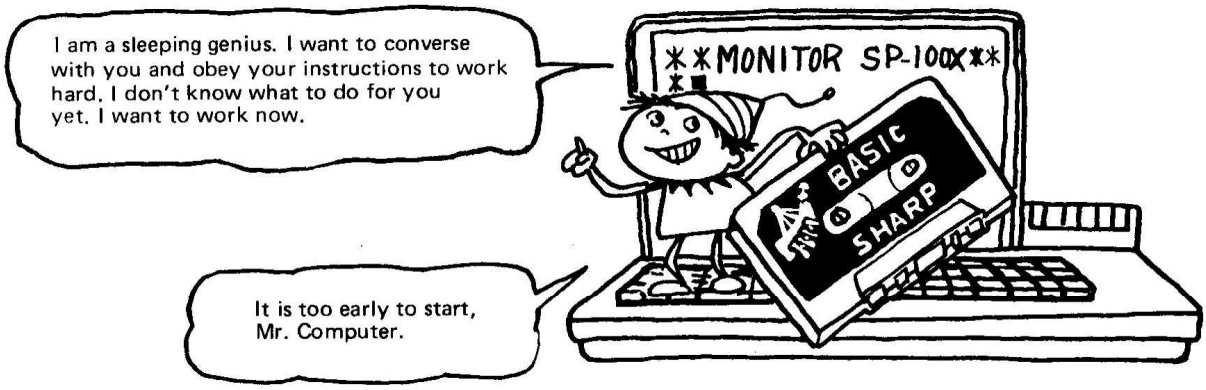
Even the Computer Memorizes Basic!

Now, begin operating your computer. First, turn on the POWER switch on the back of the machine. The following will be displayed on the TV screen.




```
** MONITOR SP-100X **
*■
```

The ■ must be flickering. This flickering ■ is termed a "cursor" on which the characters or symbols representing a key appears when a key is depressed. The cursor shifts to the right, and is in its waiting condition for the next key-in. In this status, the computer is in a position to do nothing, for it has not yet memorized a common language to converse with you. What it can do however in this status is to memorize such a common language.



It usually takes a long time for human beings to memorize a working language, and requires a lot of effort. Contrary to this, the computer can do the same job quickly with simple operations. Teaching such a language to the computer from a cassette tape is called "loading".

Press the keys by selecting the characters on the keyboard in the following order.

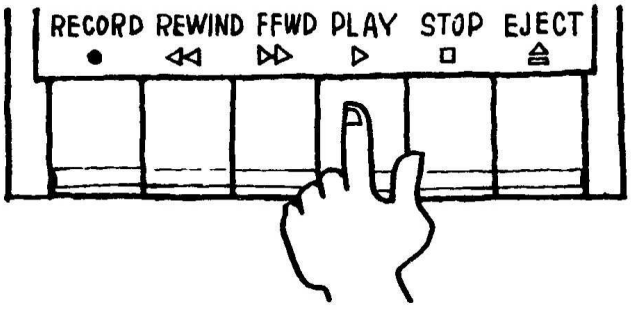
[L] → [O] → [A] → [D] → [CR]

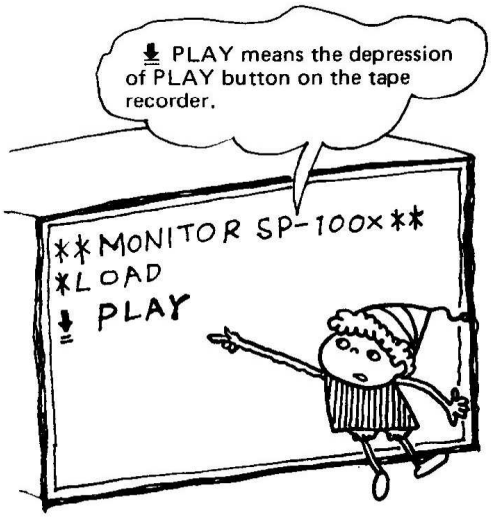
If you make a miss-key operation,

> [INST · DEL] key is to be pressed to delete the character for the key-in of a correct character, or
> [CR] key is to be pressed for the operation all over again.

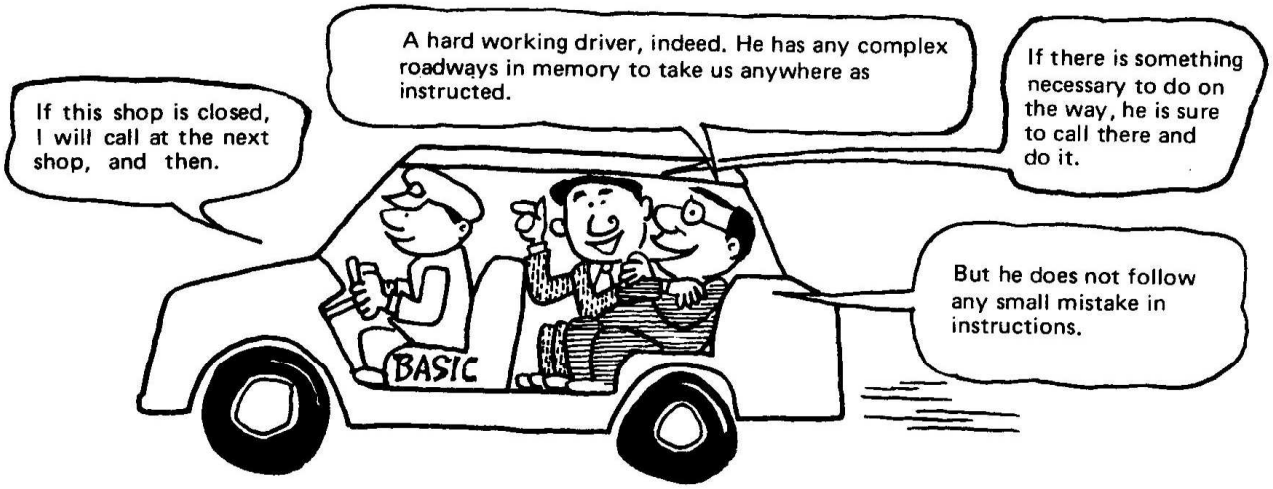
You have done it well, haven't you? With correct key-in, the display on the TV screen is as illustrated below. With this, the computer is ready to receive a common language into memory.

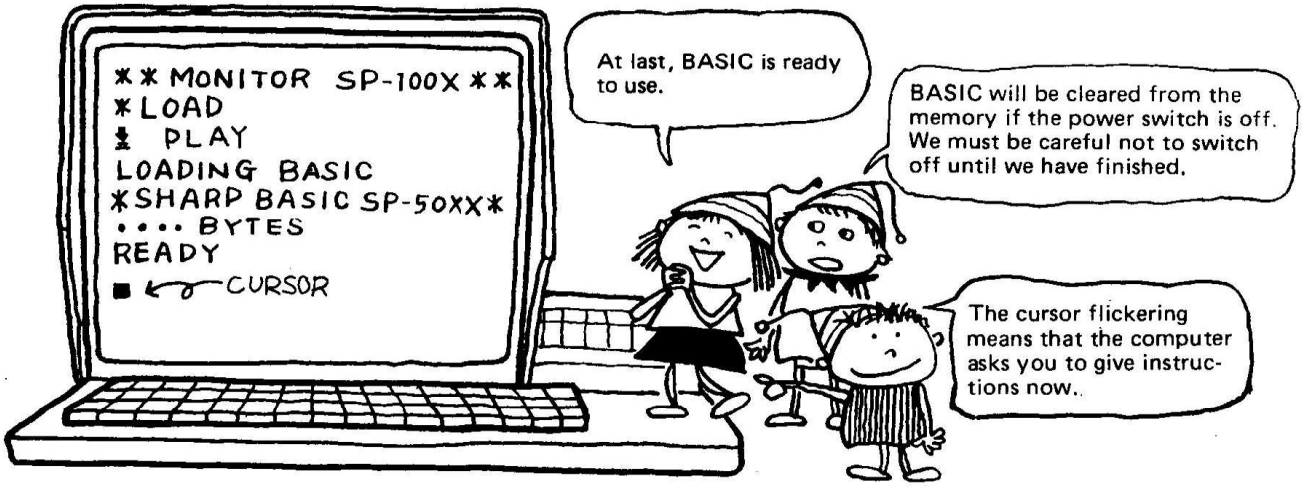
Then, press the PLAY button of the tape recorder to let the computer memorize the BASIC language.





BASIC is Fond of Conversation





The above has made the computer memorize the BASIC, ready to execute your instructions. Now is the time for you to use the BASIC. That's right, the time for your learning.

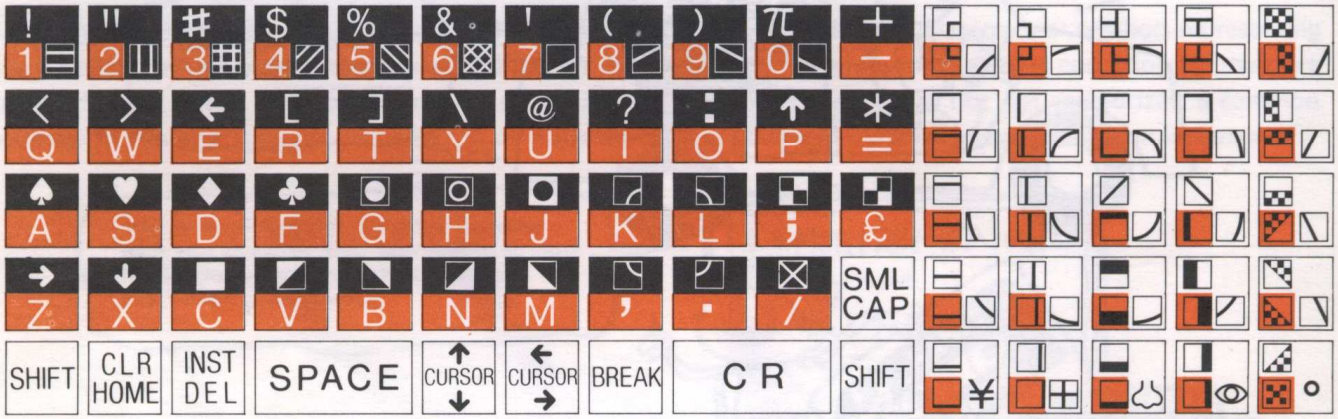
With the computer before you, operate it to learn the BASIC. At the beginning, try a number of operations to make your learning easy. The computer will never be broken with any key operation. Try pressing keys, to see what the computer answers. If a repeated action loop occurs during operation, press the BREAK key while the SHIFT key remains depressed. The computer stops and returns to its READY condition waiting for a new instruction.

How to Use the Keyboard

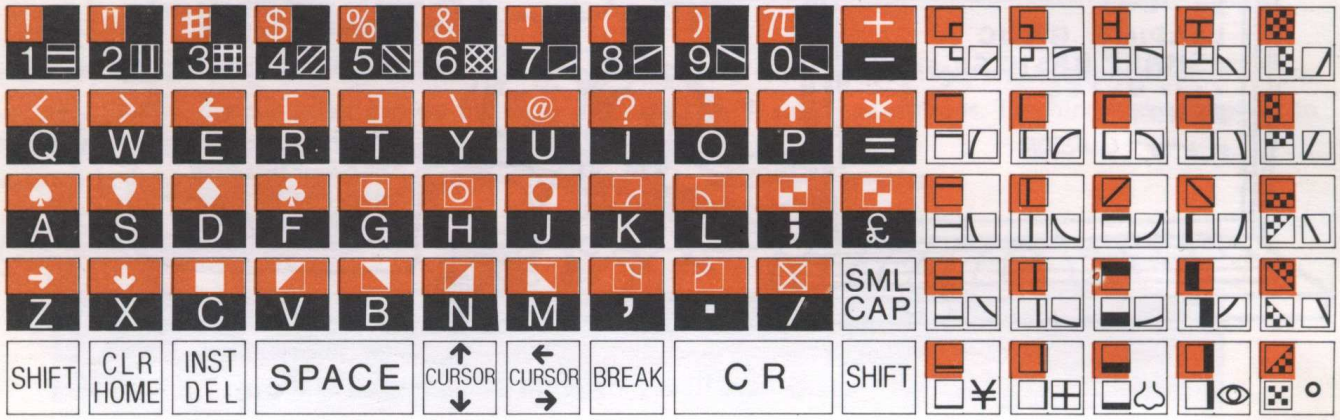
The keyboard is a means for you to converse with the computer. The keys are separated by colours, and classified into two groups depending on the use, as follows:

Black and blue keys ..... These are for keying-in of a total of 204 characters and symbols.
Yellow keys ..... These are for function controls.

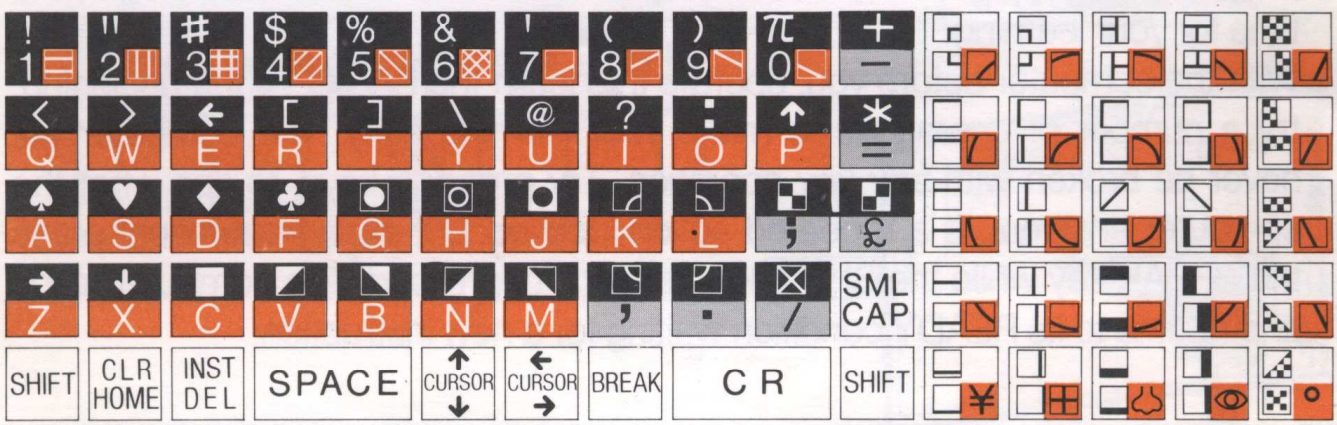
(1) With character and symbol keys pressed, the characters and symbols in the following coloured portions are displayed on the TV screen.



(2) While pressing the SHIFT key in the yellow key group, press the character and symbol keys to display the characters and symbols in the coloured portions below.

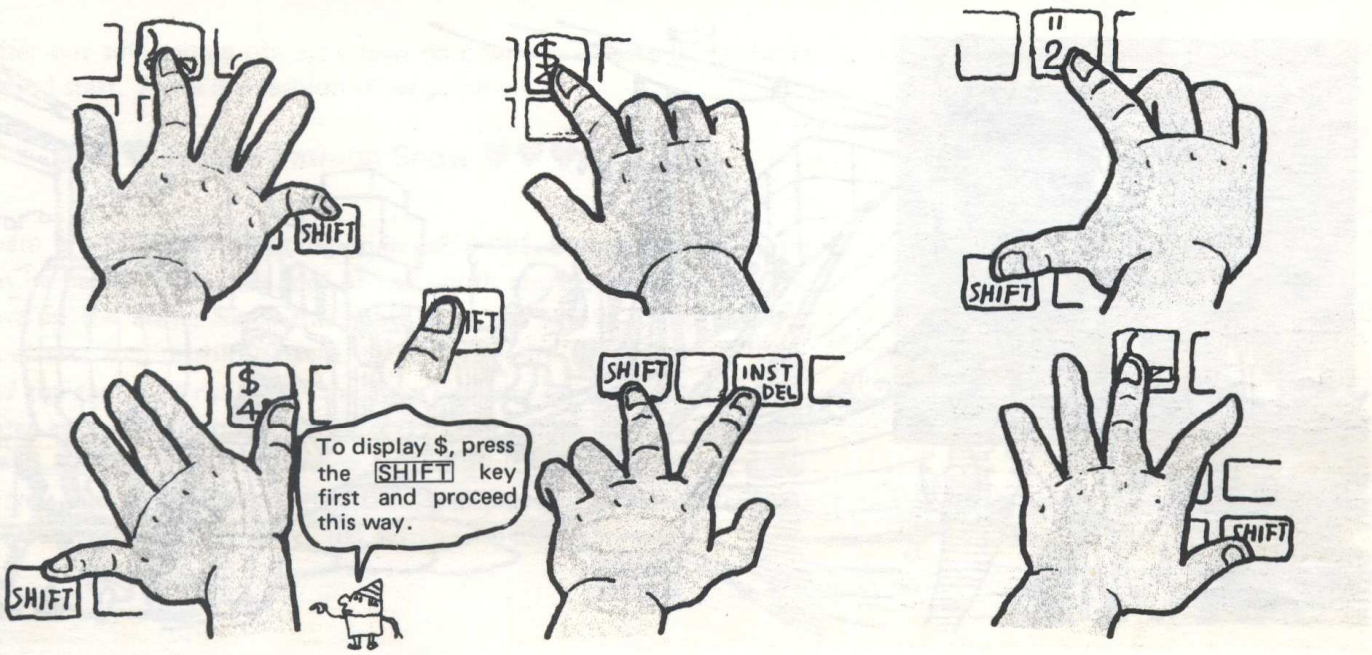


(3) While pressing the SHIFT key in the yellow key group, push down the SML · CAP key, and the lamp changes from green to red. Pressing the character and symbol keys afterwards results in display of the characters and symbols in the following coloured portions. However, the alphabet will be displayed in small letters although they are illustrated here in large letters.



To display characters and symbols shown in (1) again, press the SML · CAP key once again. The lamp light changes from red to green to return to status (1).

Yellow Keys are Magicians



Special Key Functions

CR

After the transfer of the instruction displayed on the TV screen to the computer, the cursor shifts to the head of the next line. CR: Abbreviation of carriage return.

CLR HOME

The cursor shifts to the top left corner of the TV screen.

INST DEL

This deletes the character at the left of the cursor, shifting characters on the right to the left by one character space. The right end becomes blank. DEL: delete

↑ CURSOR ↓

The cursor shifts down by one character space. When the cursor is at the bottom end of the TV screen, the display is shifted up by one line. No character is cleared even if the cursor passes through it.

← CURSOR →

The cursor shifts to the right by one character space. When the cursor is at the right end, the display shifts to the left end one line down. No character is cleared even if the cursor passes through it.

BREAK

Stops read or write operation when this key is pressed while reading or writing a cassette tape.

SHIFT

With the SHIFT key depressed, pushing the special keys down changes their functions as follows:

CLR HOME

This deletes the display and the cursor shifts to the top left corner of the TV screen. However, the program and parameter contents remain unchanged. CLR: clear

INST DEL

This inserts a blank in the cursor position, shifting the characters on the right to the right by one character space. INST: insert

↑ CURSOR ↓

The cursor shifts up by one character space. Positioned at the top of the screen, however, the cursor remains unmoved. No character is cleared even if the cursor passes through it.

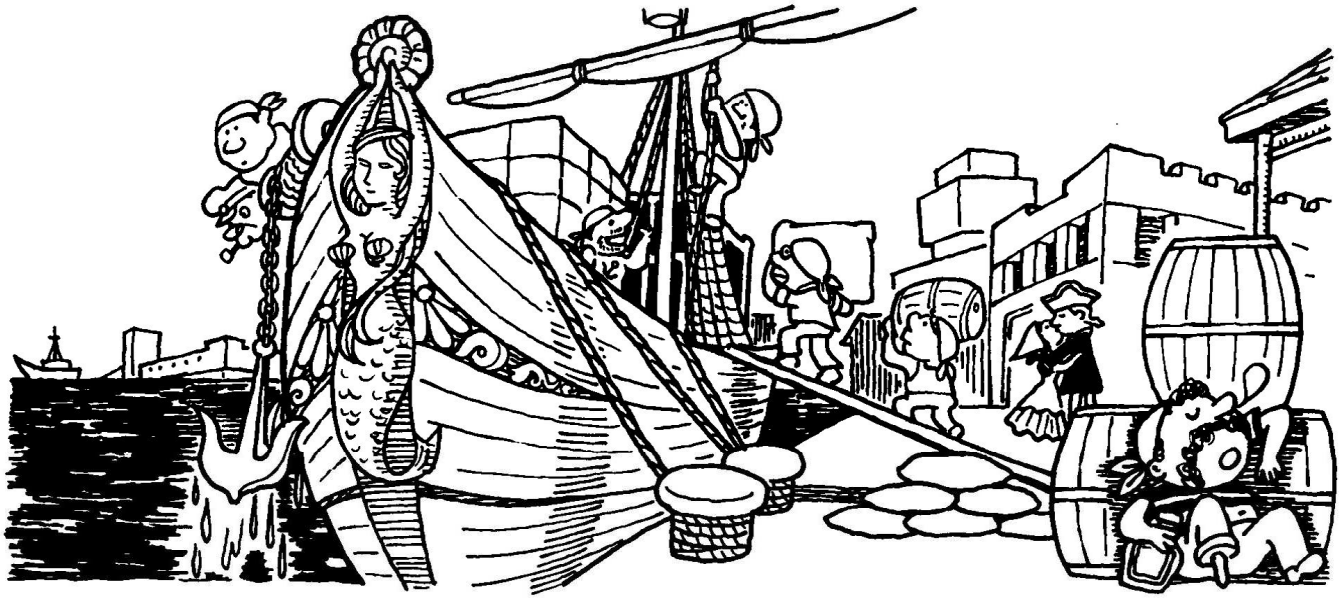
← CURSOR →

The cursor shifts to the left by one character space. When positioned at the left end, the cursor shifts to the right end one line above. No character is cleared even if the cursor passes through it.

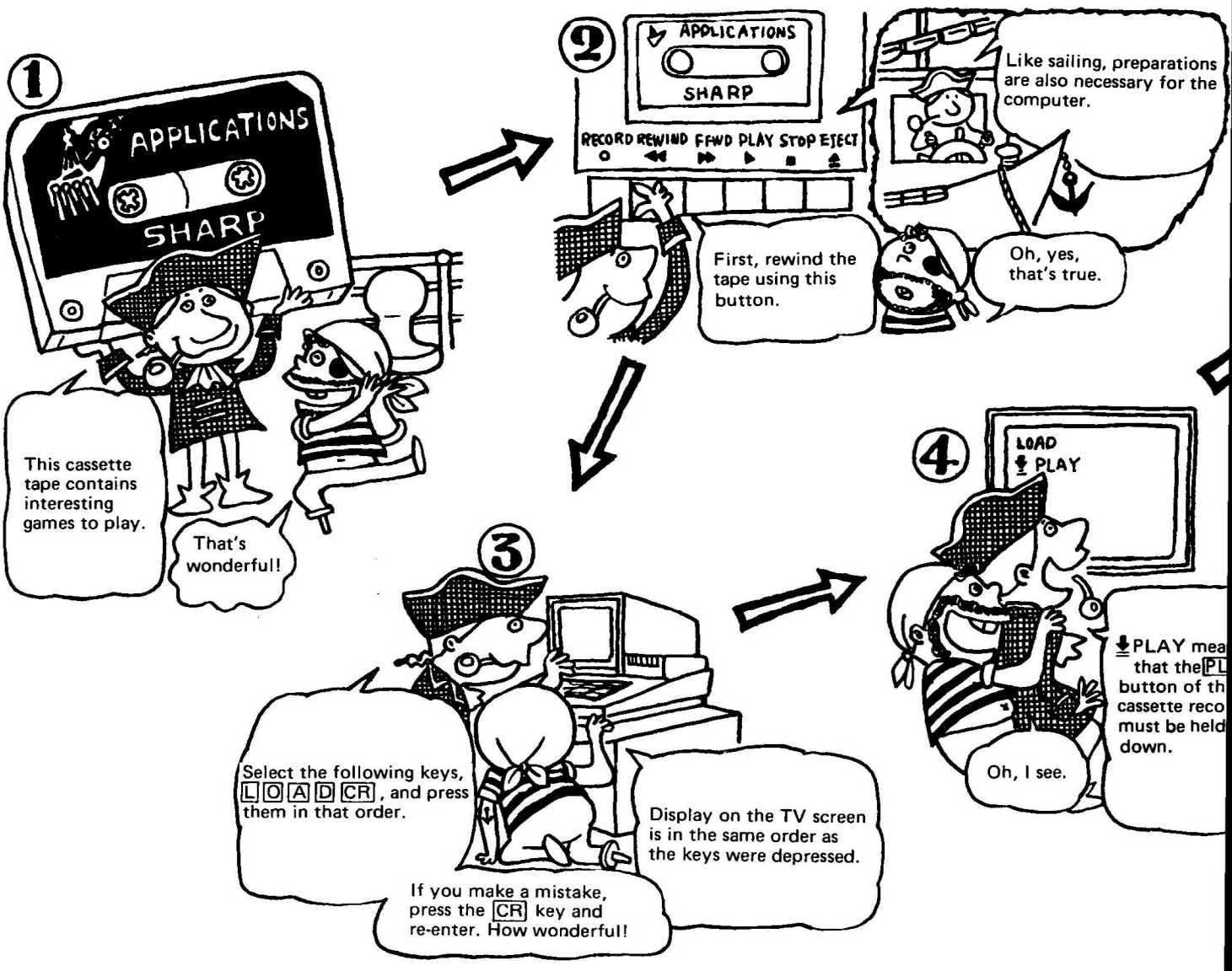
BREAK

Stops program execution.

Sailing Now with BASIC



Now, start on a fascinating voyage. First, let's have a sailing ship for us to be on board. The ship is there in the cassette tape marked "APPLICATIONS". Similar to the BASIC reading, reading should be done in the following procedure. Best learning is trying without hesitation.

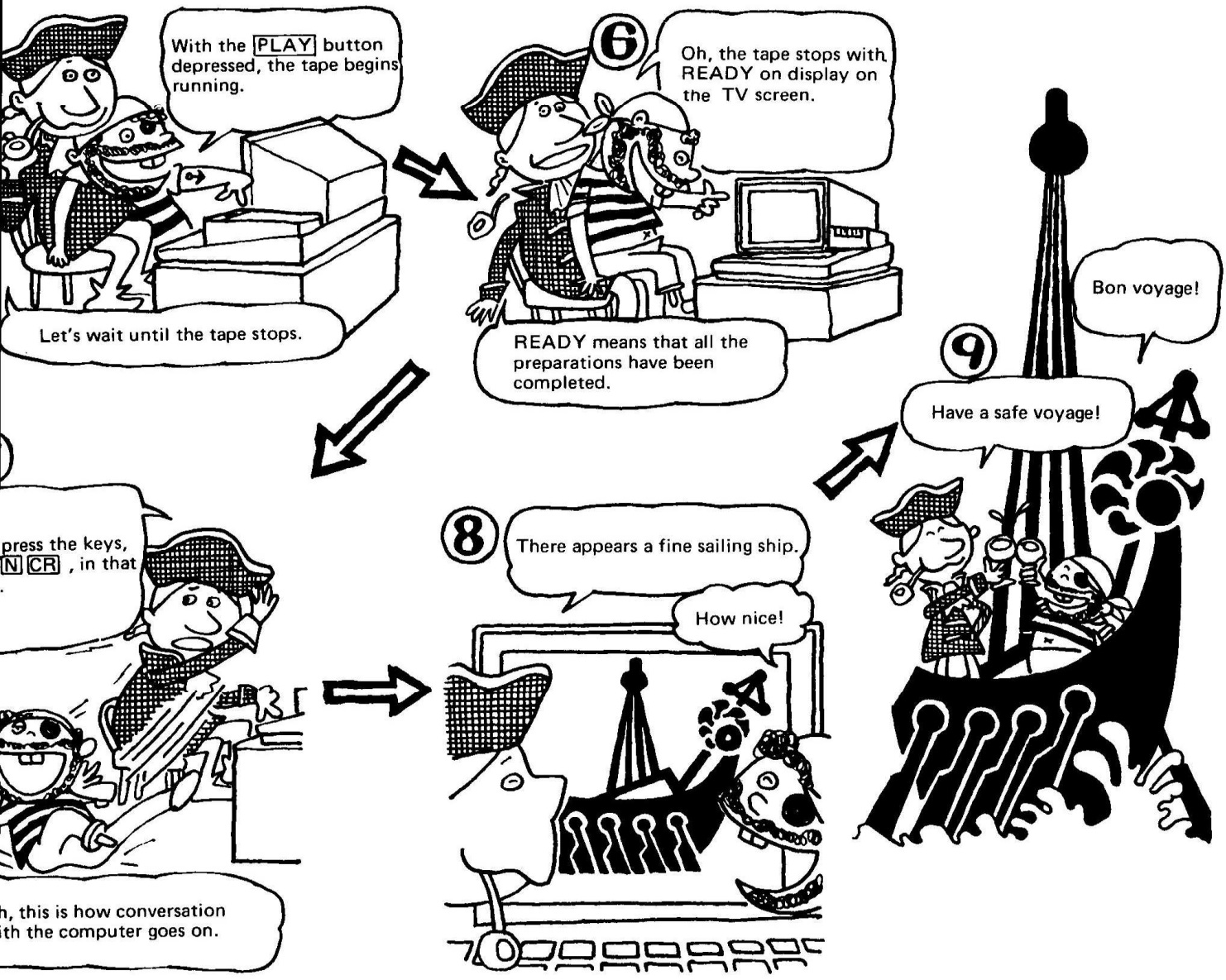


Ellice's Fashion Show

After our safe sailing off, let's now go down to a party to celebrate a good start. There is a fashion show going on.

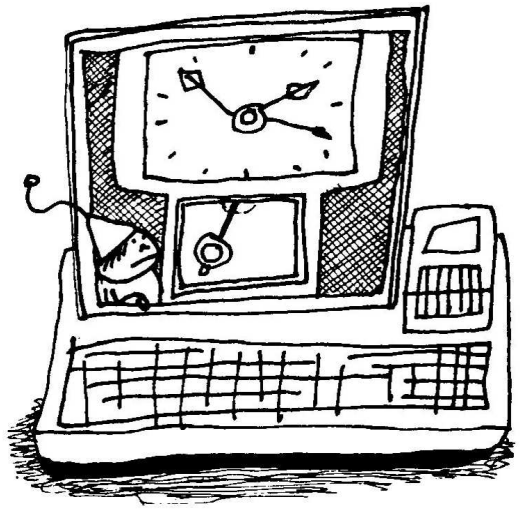
♥♥♥ Ellice's Fashion Show ♥♥♥

There is a fashion show going on in cabin 101. Something like music can be heard. Ellice, the idol of the youth aboard the ship Argo, is now on the stage. Since this show is on the side A of the cassette tape next to the ship pattern, similarly key-in [L] [O] [A] [D] [CR], and the tape will run. With the tape recorder set to STOP, the computer askes you to press the [PLAY] button, and you can hold down the [PLAY] button. After loading, key-in [R] [U] [N] [CR] in that order as you did before when READY is displayed on the TV screen. With Ellice and music, cabin 101 is in a very happy mood.



Internal Clock and Take-Down Game

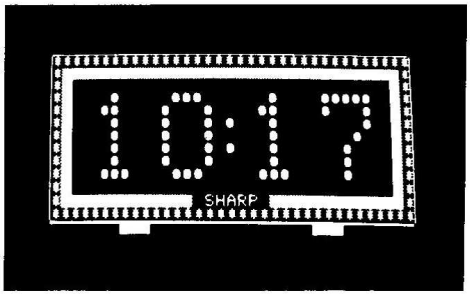
The computer has an internal clock.



With the power switch on, this clock starts counting time up, instantly setting the time to 00 hours 00 minutes 00 seconds automatically. The 00 has a meaning. The clock inside the computer consists of two figures each for hours, minutes and seconds, making a number of 6 figures.
How many minutes have passed since the power switch on the computer was turned on? If it was 30 minutes ago, the number of 6 figures must be 003000. In the same procedure as before, load the clock program to be described below.
The cassette tape restarts and stops, then the computer displays READY on the TV screen to indicate that loading has been completed. Then, set the internal clock to the actual time of your watch. For example, when your watch shows 10 hours 35 minutes 12 seconds A.M. set the internal clock at 10 hours 36 minutes A.M.. This requires the following keying in operation: With

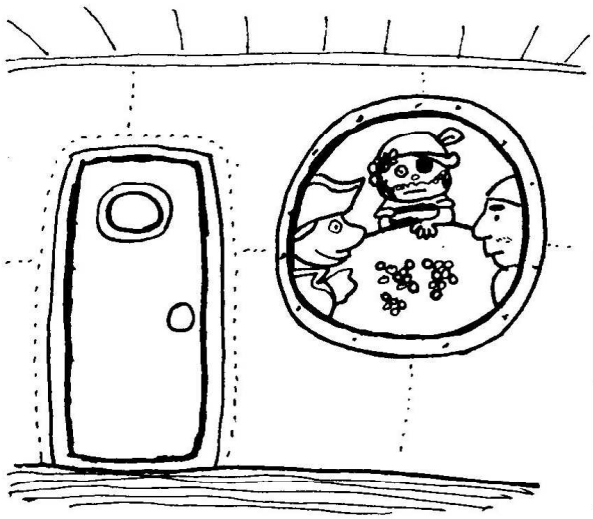
[T] [I] [$] [=] ["] [1] [0] [3] [6] [0] [0] ["]

keyed-in, press the [CR] key simultaneously with the actual time of 10 hours 36 minutes when it comes, and the internal clock is set. All you do after this is RUN.



In the cabin 102, happy sailors enjoy a take-down game. Now, let's start loading this program. How? The procedure is the same as we have done before. The rules of this game are simple:

- ★ You and the computer take a stone or more alternately from a pile of stones.
- ★ The pile becomes smaller in size, and the last stone is taken to win the game.



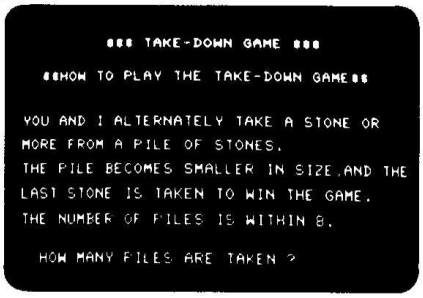
.....Then Going onto a Program World

Have vou finished program loading?
Now, key-in RUN to start the game.

[R] [U] [N] [CR] . Don't forget to press the [CR] key.

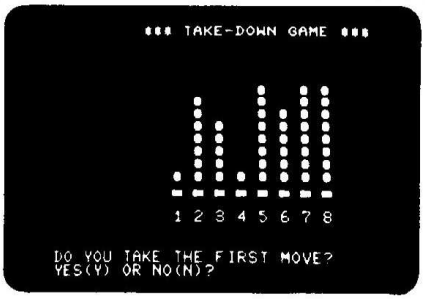
Now is your turn to think. You will lose this game unless you think.

1



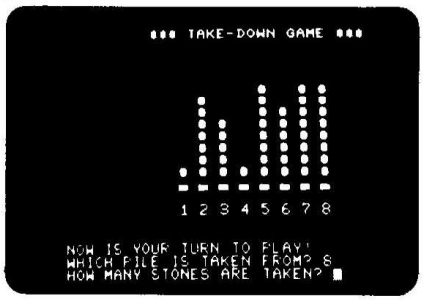
The computer will ask a question of how many piles are made.
To make 3, for example, key-in [3] [CR] . The number of piles can be assigned from 1 to 8.
The computer then asks if the number of piles is correct. If answer is yes, key-in [Y] [CR] . If answer is no, key-in [N] [CR] and re-enter number of piles.

2



The computer will then ask a question of who will take the first move. To take the first, key-in [Y] [CR] . To take the second move, key-in [N] [CR] .
Now, key-in [Y] [CR] to take the first move.

3



Start the game!
The 8 mountains are now on display. The computer will ask a question of which pile is taken from. For example, answer [8] [CR] to the question.
When asked how many stones are taken, key-in the number of stones to be taken. For example, when your answer is [2] [CR] , you take 2 stones from the 8nd pile.

What are the Computer Functions Described so far Based On? What is there on the Cassette Tape?.....

This is the sea you have sailed to; The World of BASIC Programs. Separate programs are generated for inclusive use in the game and the clock. Here, let's have a look at what a program is like. Press the keys, as shown below:

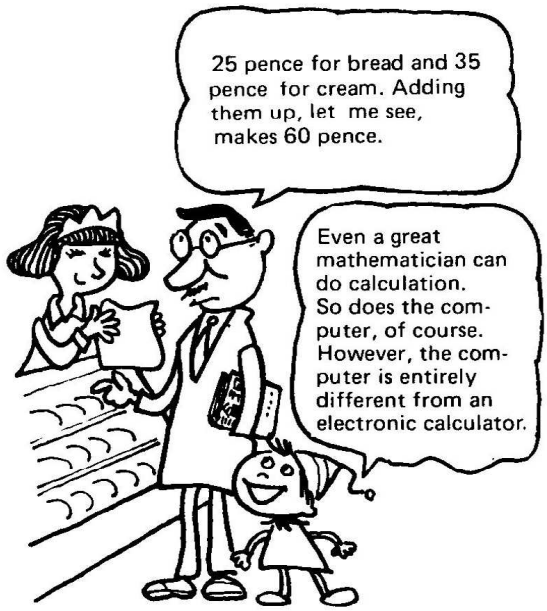
[L] [I] [S] [T] [CR]

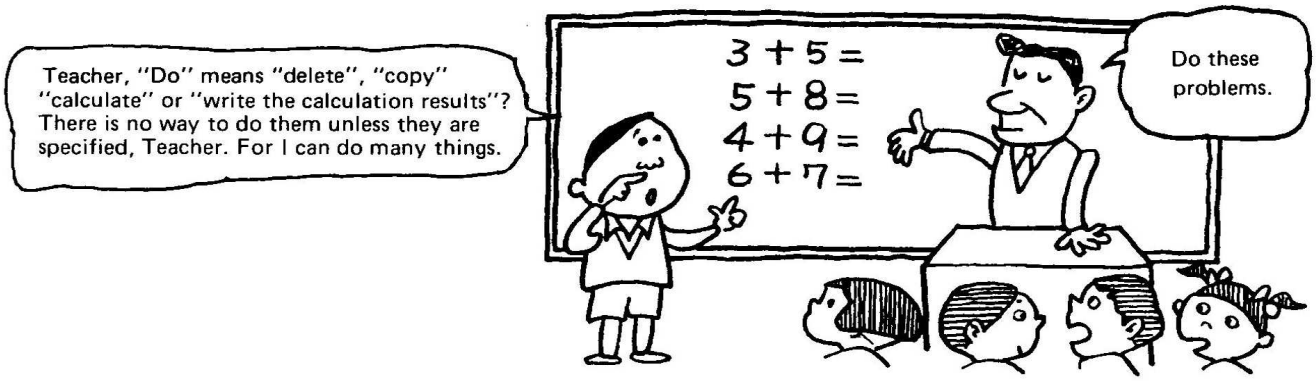
What has been displayed on the TV screen is the program for the take-down game. Your ship is now turning slowing to head straight for its destination.

That's right!.....Your destination is to generate or modify a program by yourself.

What is the Direct Mode?

Using the computer like an electronic calculator is possible if required. This kind of operation is called "Direct Mode".
Like the electronic calculator, key-in 5 + 8=.
To key-in the [+], press the key while holding down the [SHIFT] key.
In fact, however, the computer displays the characters on the TV screen only as keyed-in, and of course, no calculation is executed even with the [CR] key depressed. Here lies the difference between your computer and the electronic calculator. Your computer requires an instruction of what should be done about 5 + 8 =.





PRINT

To use the computer in the same manner as the electronic calculator, the computation of 5 + 8 is required to be displayed on the TV screen. For this, there is the PRINT command available as an instruction. Using this command, let's press the keys in the following order to transfer the instructions.

[P] [R] [I] [N] [T] [5] [+] [8] [CR]

As the keys are depressed, the characters below will be displayed on the TV screen.
How about your computer?

READY ..... Meaning "Go ahead with your work".
PRINT 5 + 8 ..... Display the computation of 5 + 8, and with the [CR] key. Pressed indicating the end of a command.
13 ..... This is the executed result of the command.
READY ..... What is to be done next?
■ ..... Cursor

The Four Arithmetic Operations are, of course, Possible

Why not do more calculations using the PRINT command?.....Calculations of a number of many figures, repeated additions, and subtraction as well.....

If you want to go on to multiplication and division, note that the computer uses signs slightly different from those of ordinary mathematics.

Multiplication sign	✳	**This is called "asterisk".**
Division sign	/	**This is called "slash".**

Calculation with Parenthesis

The computer is capable of handling more complex calculations than an ordinary calculator. This is a calculation with parenthesis.

In case of ordinary mathematical operation, different signs of groupings are used to write as follows:

$3 \times 6 \times [6 + 3 \times \{9 - 2 \times (4 - 2) + 1\}]$

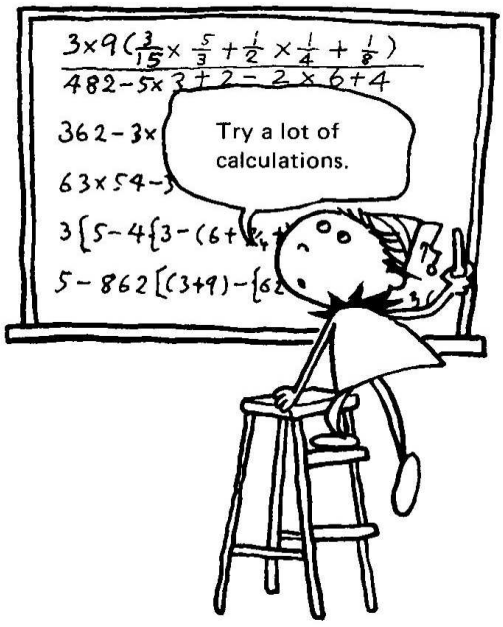
Whereas the parenthesis ( ) alone is used at all times with the computer.

3✳6 (6 + 3✳(9 − 2✳(4 − 2) + 1))

Even with the above, the computer never forgets the rule that computation in the inner signs of groupings be done first, and never makes any mistake.

Exercise

```
PRINT  (6 + 4) / (6 − 4)
 5
PRINT  3✳(5 + 9✳(9 − 2) − 6 / (4 − 2) ) + 5
 200
PRINT  (3 + 4)✳(5 + 6)
 77
PRINT  (10 + 20) / 6✳(2 + 3)
 25
PRINT  (10 + 20) / (6✳(2 + 3))
 1
```



String? Equation?

```
PRINT 3 + 5
```

With the above, pressing the CR key makes 8, doesn't it? Now, put the equation in quotation marks".

```
PRINT "3+5" and CR
3+5
```

Oh, the result is different. Try another one.

```
PRINT "HELLO MY FRIEND" CR
HELLO MY FRIEND
```

As is clear from the above, the characters or symbols put between quotation marks " are displayed as they are on the TV screen.
The block of characters and/or symbols between the quotation marks is called a **string.**

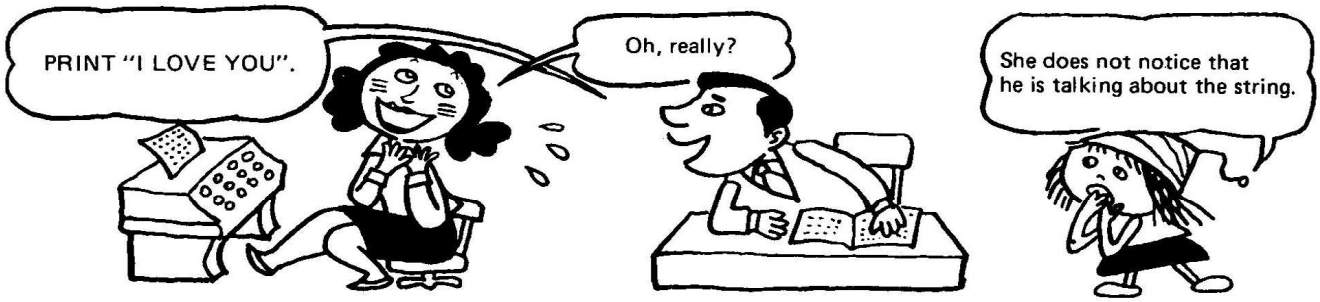
PRINT "3+5"

This is a string put between quotation marks.

PRINT 3+5

This an equation, not a string.

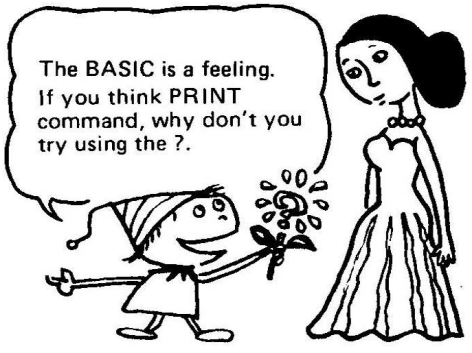
It is necessary for you to know more about the strings. The free use of strings will double the pleasure in operating the computer.



PRINT ???

This is the command which you will have to get along with quite often. If you think it troublesome to key-in PRINT at every operation,
press the ? in place of PRINT.
The computer automatically converts the ? to PRINT.

```
? 3 * 5
 15
? (3 + 4) * 10
 70
```



What are the PRINT's 1st and 2nd Approaches?

t is possible to add a plurality of items, such as strings and equations, to the PRINT command. In this case, ndividual items should be separated using semicolons and commas.

```
PRINT "3+5 = "; 3 + 5 CR
3+5=  8
```

Semicolon

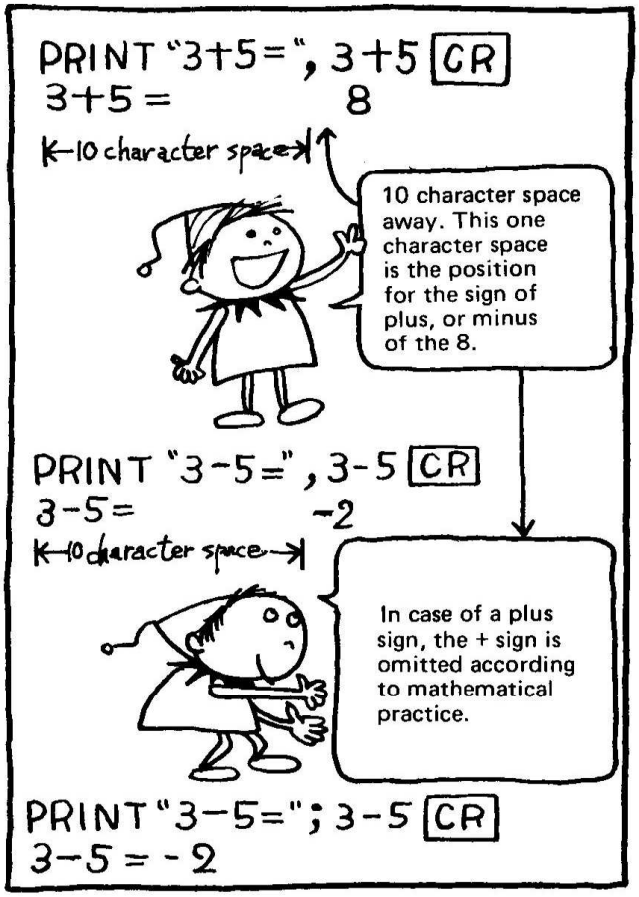
he equation between the quotation marks is a string. he actual calculation is done according to the equaon following the semicolon.
ry it.
/hat will happen when using a comma (,) in place of semicolon (;)?

```
PRINT "3+5=", 3+5 CR
3+5=          8
```

comma

/hy! The result of 3+5 is displayed far away from ne equation. This means the following difference lies etween the semicolon and comma.

; ..... Display is made next to the equation.
, ... Display is made 10 character space away from the equation.



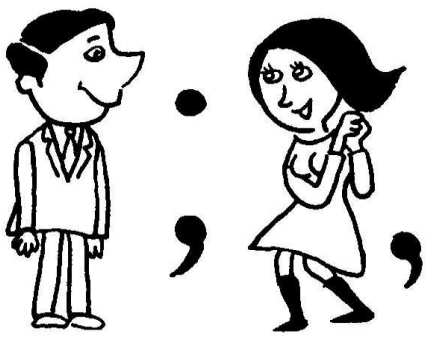
/hen a separation is made with a comma, the 6 character space is not away from the end position of a string, but 0 character space from the starting position of the string. This fact requires your special attention.

```
PRINT "12345", 3+5 CR
12345      8
```
10 character space

```
PRINT "123456789", 3+5 CR
123456789  8
```
10 character space

f the string is longer than 10 character space, the figure 8 is automatically made a further 10 character space way.

```
PRINT "123456789012", 3+5 CR
123456789012          8
```
20 character space

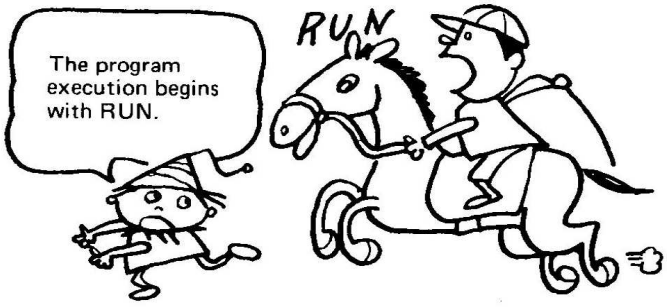


2nd approach

Let the Computer Run!

Here is a program with statements covering several lines.

```
10 A=3
20 B=5
30 C=A+B
40 PRINT A, B, C
50 END
```



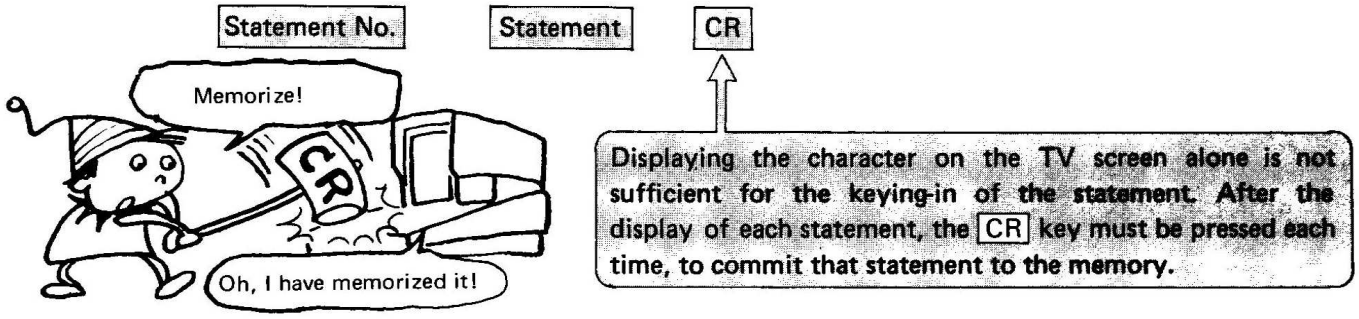
This program requires no special explanation, does it?
This program is the one to instruct calculations of A=3 and B=5, and the display of A, B and C on the TV screen in equation as follows:

C=A+B

The numeral at the head of each statement is called a statement number. The computer is sure to execute the statement numbers from small in value to large in the correct sequence. Therefore, this makes it possible to insert a new statement in the program afterwards. For example,

35 D = B – C

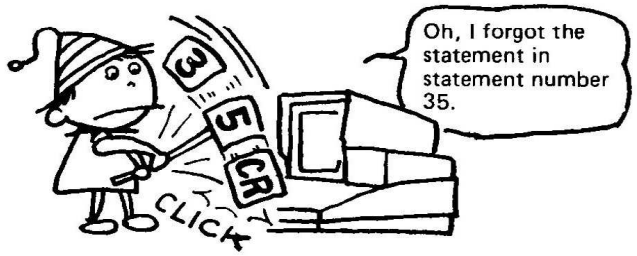
The computer executes a program in the sequence of the statement numbers, and therefore, the statement numbers are made in steps of 10, as illustrated in the above example, so that new statements can be inserted later whenever required. The statement numbers can be selected at liberty from 1 to 65,535.



For example, presuming the following,

35 CR

the statement in statement number 35 is deleted from the computer.

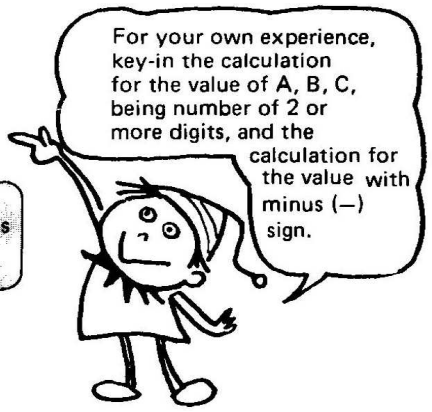


Now, let's execute the program.

Press the keys as follows RUN CR .

```
RUN
 3                    5                    8
←— 10 character space —→←— 10 character space —→
```

One character space for plus or minus sign. For minus value, minus (–) is inserted.

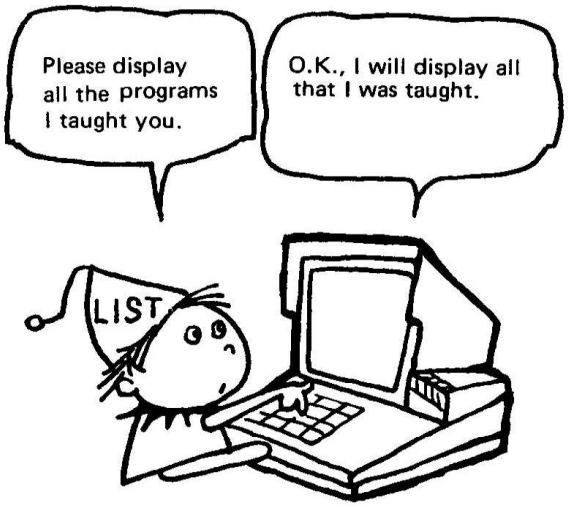


List for Quick Understanding

While continuing conversation with the computer by repeated trials and errors, the first keyed-in program may sometimes be gone from the TV screen. Even so, don't worry. The computer never forgets any program once keyed-in. When you want to see the previous keyed-in program, key-in the following:

LIST CR

This is followed by the display of all the stored programs on the TV screen. If the program extends over tens and hundreds of lines beyond display at a time, part of the stored programs can be displayed.



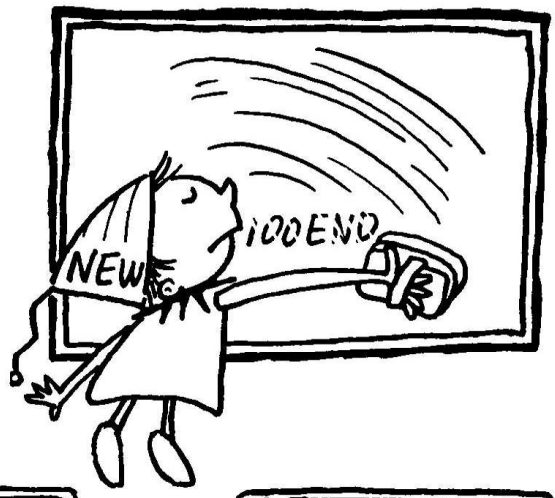
LIST –30 CR	Displays a program up to statement number 30.
LIST 30– CR	Displays a program after statement number 30.
LIST 30–50 CR	Displays a program between statement numbers 30 and 50.
LIST 30 CR	Displays a program of statement number 30.

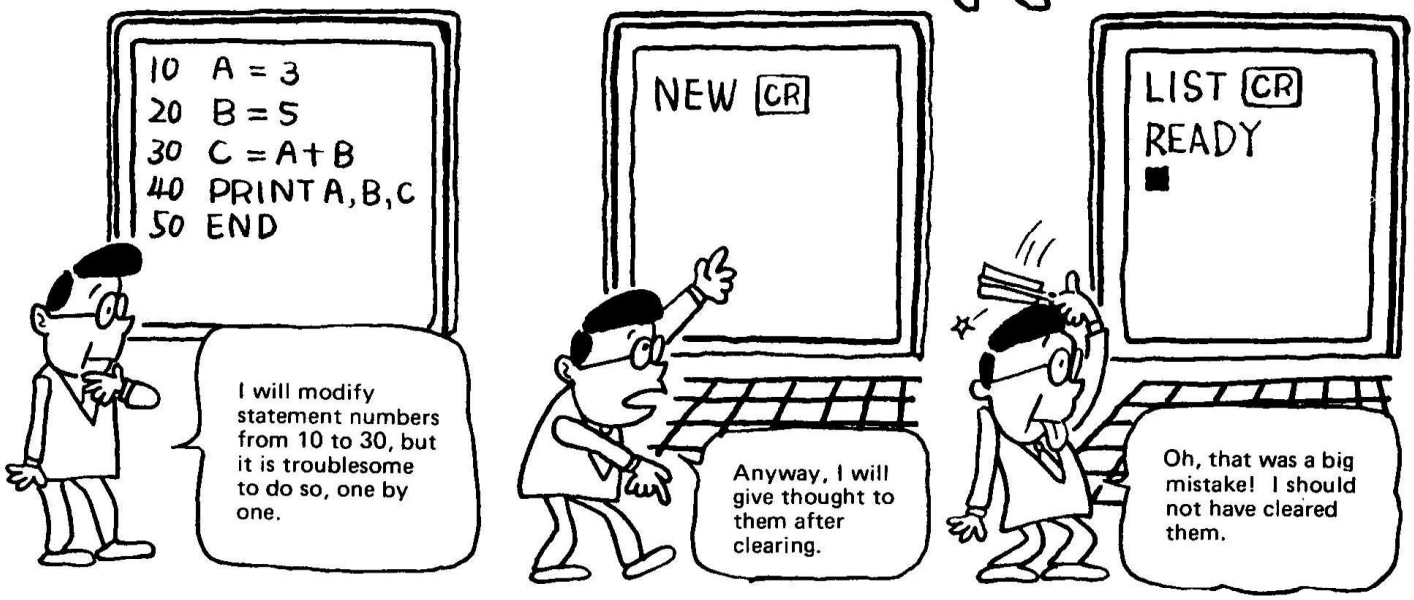
The result of NEW .....

To store a new program, clear the previous program using the NEW command. Otherwise, two programs may overlap to cause confusion.

NEW CR

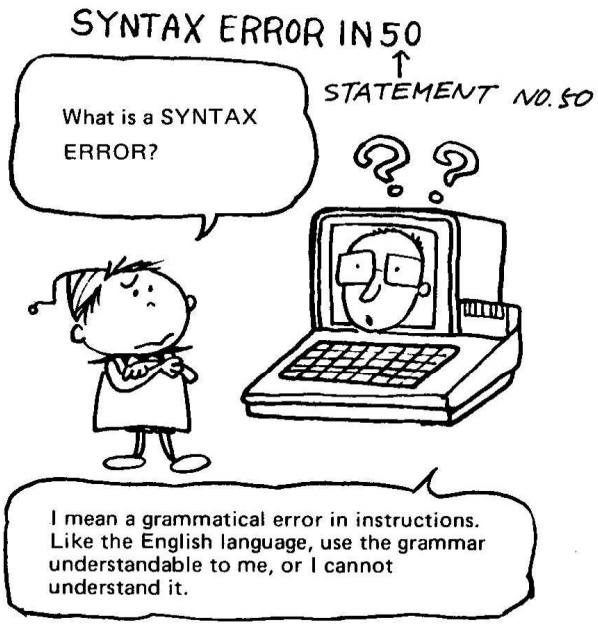
This will clear the previous program completely. To ensure this, key-in the LIST command to check that the program is cleared.





Error Puts the Computer in Confusion

```
10 A=3
20 B=5
30 C=A+B
40 PRINT A, B, C
50 END
```



This is the same program as used before. Did it run well? If there is an error in any statement, the computer tells you about it. For example,

50 EMD

If you make a mistake of M for N, the computer executes the program up to statement number 40 as instructed, but it does not know what EMD is all about. The computer tells you about a syntax error, as follows:

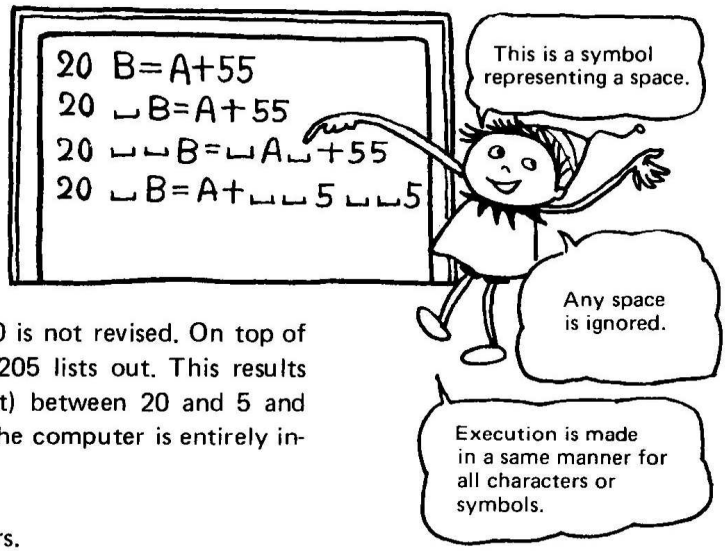
SYNTAX ERROR IN 50

Then, key-in correctly as 50 END. For two statements identical in statement number, if any, the computer takes up the one that was keyed-in later. With this, is your program complete?
If so, try to make a mistake in statement number 20, for example.

20 5 = B

With this, the statement in statement number 20 must be revised.
Sure?.....Use the LIST command to check the revision.

```
10 A = 3
20 B = 5
30 C = A + B
40 PRINT A, B, C
50 END
205  = B
```



Oh, something funny occurs. Statement number 20 is not revised. On top of that, a strange statement with statement number 205 lists out. This results because the computer ignored a space (blank part) between 20 and 5 and arranged them as a statement number. A space to the computer is entirely insignificant and ignored.

Note : See page 114 for detailed information of errors.

Collect the Statement ¿¿¿

f you want to do the following about your program which has been stored in the computer;

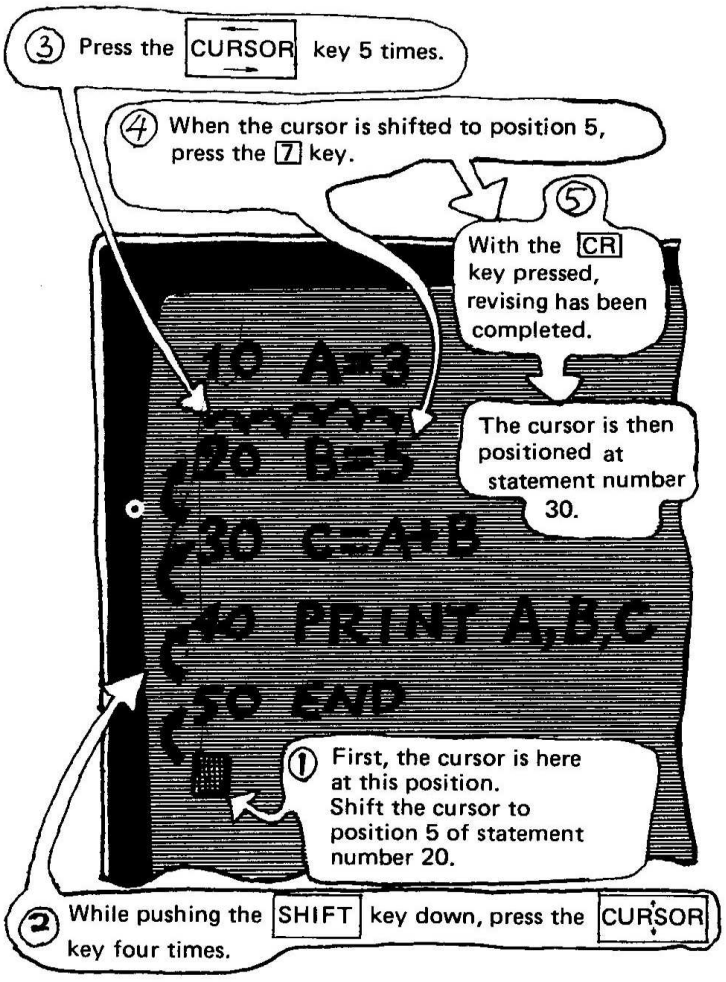
To correct errors,
To modify for a better statement,
To modify for a separate statement,
To modify part of a complete statement to generate a new statement,
.....

et's study about statement modification, insertion and deletion when the above are required.

Cursor Shift

o revise the characters in a statement, the cursor ust be shifted to the respective character positions. ow, let's revise 5 of the statement B = 5 in statement umber 20 to 7. Refer to the diagram at right for the ift procedure.

ith this, the program displayed on the TV screen as been modified. In fact, however, this has not yet odified the program stored in the computer. To odify the stored contents, the CR key must be essed. What? Did you key-in 6 instead of 7? o modify the character to the left of the cursor, ere are two methods available.



Method 1 Pressing the INST·DEL key.
With the INST·DEL key held down, the cursor shifts to the left by one character space, deleting the character next to it on the left. Press the 7 key again. Needless to say, the CR key must be pressed finally.

Method 2 Shifting to the left using the CURSOR key.
While pushing SHIFT key, depress the CURSOR key. The cursor shifts to the left by the number of times the key is pressed.
Then, press the 7 key again. The CR key must be pressed finally.

Correct the Statement!

Character Insertion

To modify the program on page 26 for the statement in statement number 30, as follows,

30 D = **100** + A + B ⇐ Do not press the CR key yet.

shift the cursor to character A. Then press the keys as shown below.
With the SHIFT key depressed, press the INST·DEL key 4 times.
There must be a space for just 4 characters to add **100** +. Key-in **100** + to this space. No more description is required for the revision of C to D. Since the statement has been modified so far, why not modify the statement number from 30 to 35, and press the CR key. Modify statement number 40 as shown below.

40 PRINT A, B, C, D

Then type RUN CR

RUN
3 5 8 108

Character Deletion

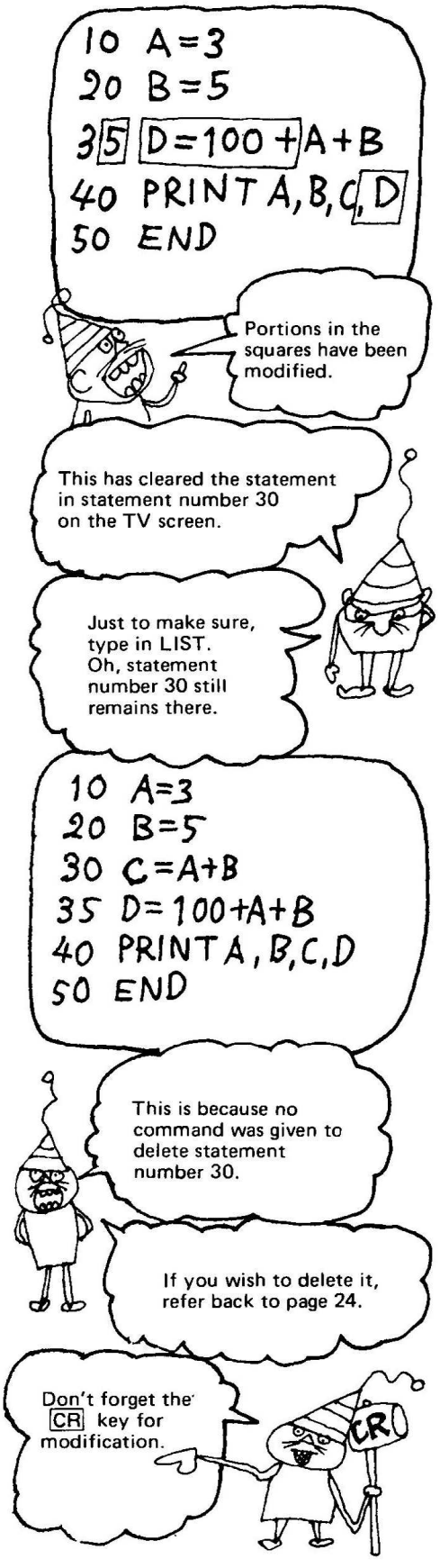
35 D = 100 + A + B

Let's modify this statement. To modify it to the following,

35 D = A + B + C

Shift the cursor to character A and press the INST- DEL key 4 times. This shifts the cursor until A + B portion comes right next to mark =.

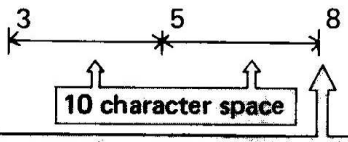
RUN
3 5 8 16



Further Study of Comma and Semicolon

For review, the following example is taken again.

```
10 A = 3
20 B = 5
30 C = A + B
40 PRINT A, B, C
50 END
```



This space before the number is for the plus or minus sign.

You remember this, don't you? In other words, when using commas between A, B and C, a numeral is displayed 10 character space away. Generate a program with new statements inserted, and run it. Statements to be inserted are the following:

```
32 D = B ↑ A
34 E = B * A
36 F = B/A
45 PRINT D, E, F
```

```
RUN
 3            5            8
 125          15           1.6666667
```

With the comma (,) revised to semicolon (;) for statement numbers 40 and 45, run the program once more. To modify the program, type in LIST and use the cursor in as smart a manner as possible.

```
RUN
 3 5 8
 125 15 1.6666667
```

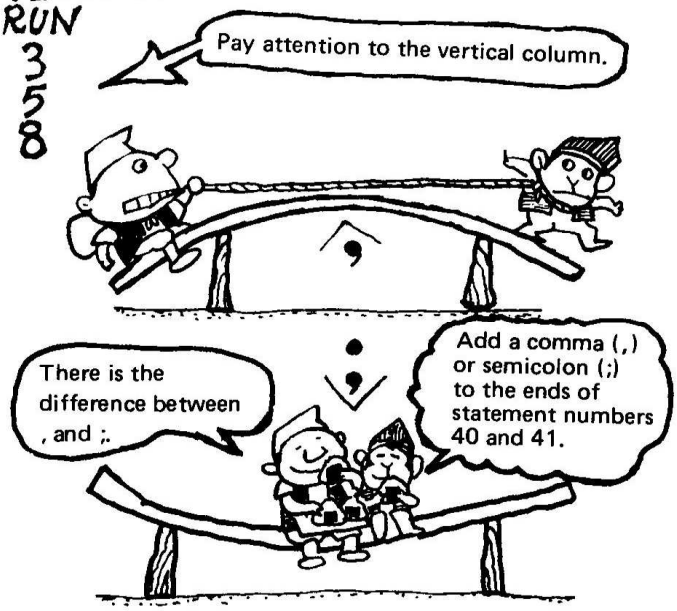
Space for plus/minus signs

Semicolon (;) has a function that combines the characters or symbols on display together. Add semicolon (;) to the end of statement number 40 then RUN, in order to make sure of this fact.

```
40 PRINT A; B; C;
RUN
 3 5 8 125 15 1.6666667
```

The following is replaced for statement number 40...

```
40 PRINT A
41 PRINT B
42 PRINT C
RUN
 3
 5
 8
```



B↑A means B to the Ath power.

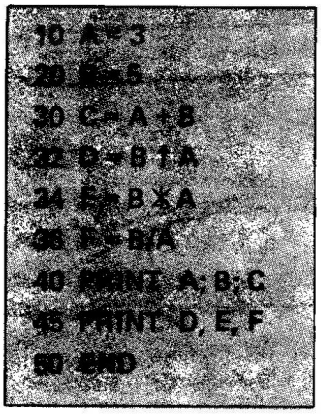
$B^A = B \uparrow A$

```
PRINT"123";"456" CR
123456
PRINT"123","456" CR
123          456
```

In case of a string its contents are displayed, irrespective of plus and minus signs.

Colon and it's use

Use of Colon



This program consists of short statements. A program in this length can be processed under one statement number, if required.


```
100 A = 3 : B = 5 : C = A + B : D = B ↑ A : E = B * A : F = B/A : PRINT
A; B; C : PRINT  D, E, F : END
RUN 100
```

Colon (:) is a symbol to be used when more than 2 statements are inserted in one statement number. This kind of statement is called a "multi-statement". A statement with 2 lines can be described in one statement number. 1 line consists of 40 characters, making it possible to use 76 characters including a statement number.

How Much Left? ..... SIZE

It is natural for you to desire to know how much storage capacity is left at your disposal as programs are stored in the computer one after another.
For this, the following is done:

```
PRINT SIZE
```

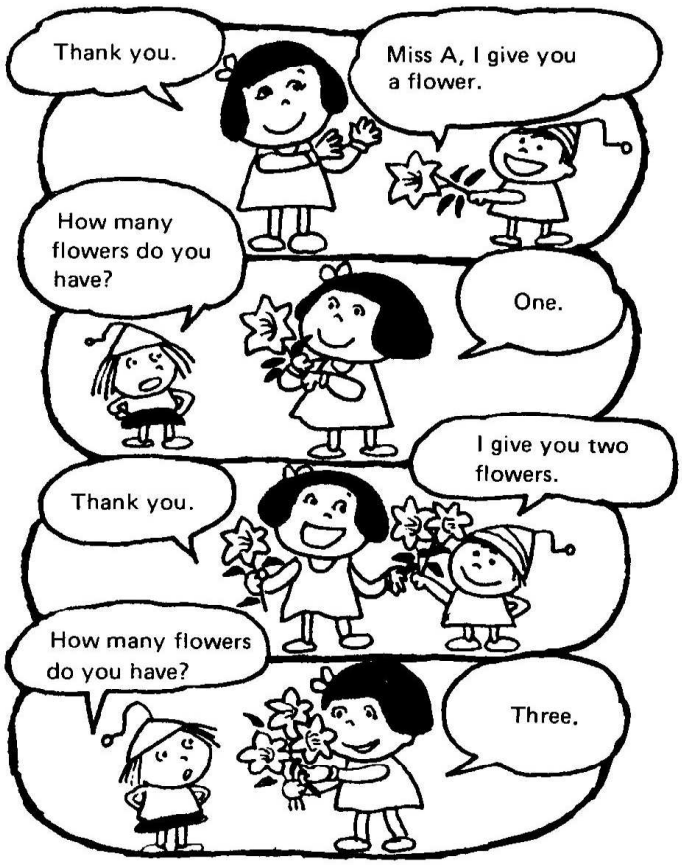
In response to this, the computer tells you about the remaining storage capacity in bytes.

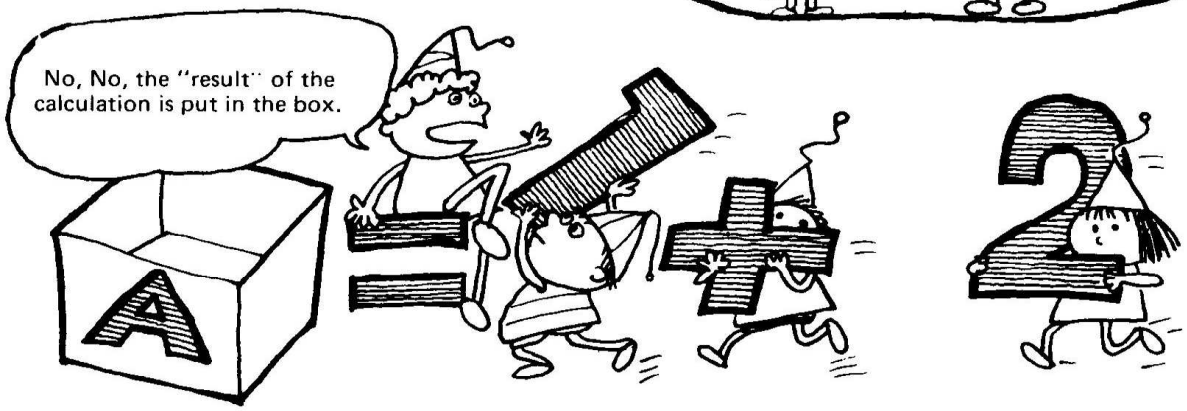
Does "A=B" Equal "B=A"?

Now, let's give attention to the = sign we have often used so far. Try the following execution.

```
10 A = 1
20 PRINT A,
30 A = A + 2
40 PRINT A
50 END
RUN
 1            3
```

1 and 3 are on display. A = A+2 is for statement number 30. If this is an equation, A is subtracted from both expressions making 0=2, resulting in a contradiction. It is not an equation.
Sign = means that the result of the right expression is substituted by symbol A prepared on the left expression.





n statement number 10, value 1 is substituted by symbol A, and at the right expression of statement number 30, the value in symbol A and 2 are added and substituted by symbol A using symbol =.
At this time, value 1 previously put in A does not exist any more.
The following 2 programs produce different results which proves that "A=B" does not equal "B=A".

```
10 A = 5
20 B = 7
30 PRINT A, B
40 A = B
50 PRINT A, B
60 END
RUN
 5            7
 7            7
```

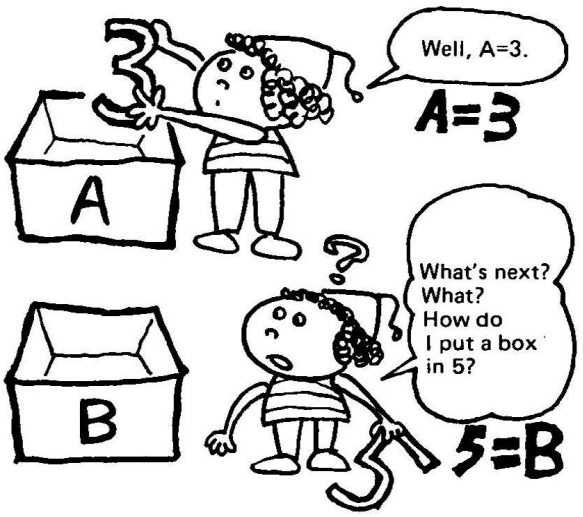

```
10 A = 5
20 B = 7
30 PRINT A, B
40 B = A
50 PRINT A, B
60 END
RUN
 5            7
 5            5
```

Variables the Computer is Very Fond of

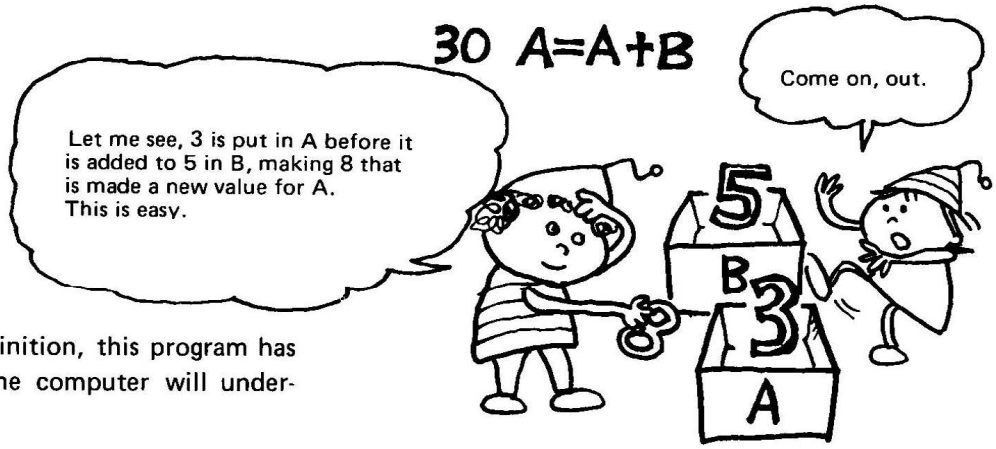
The variables used in the computer statements are different in usage from the mathematical variables. The statement-used variables are the names given to the boxes designed to accommodate values.

B = 5

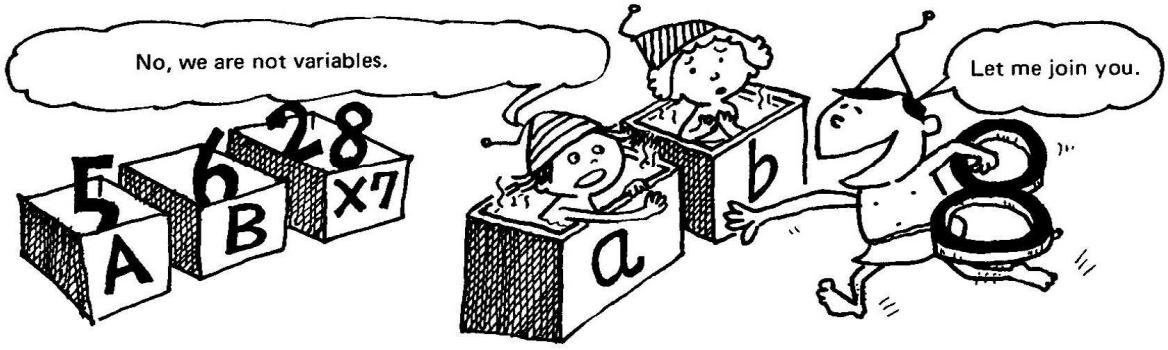
This means that value 5 should be substituted by box B. Therefore, the use described under the "Error Puts Computer in Confusion" results in a difficult statement for the computer, though it cannot be mistaken as a statement number. Because it says to put box B into 5.




```
10 A = 3
20 B = 5
30 A = A + B
40 PRINT A
50 END
RUN
 8
```



From the mathematical definition, this program has a contradiction, however the computer will understand.



Characters subject to Variables

1. A variable should be a combination of two or less characters. Any variable over 2 characters can be stored, but the characters after the second are neglected in computer processing. For example, ABC and ABD can be displayed. In processing, however, they are regarded as the same variables as AB.
2. The following are the characters for use as variables:
   (1) A to Z Alphabetical 26 ways.
   Example: A, M, Z
   (2) 260 characters with numeral of 1 figure (0 to 9) added to the alphabet.
   Example: A0, K5, Z9
   (3) Characters with two alphabetical characters combined.
   Example: AA, BK, XZ
   However, some variables, such as IF, ON, TO, etc in Basic commands, should not be used.

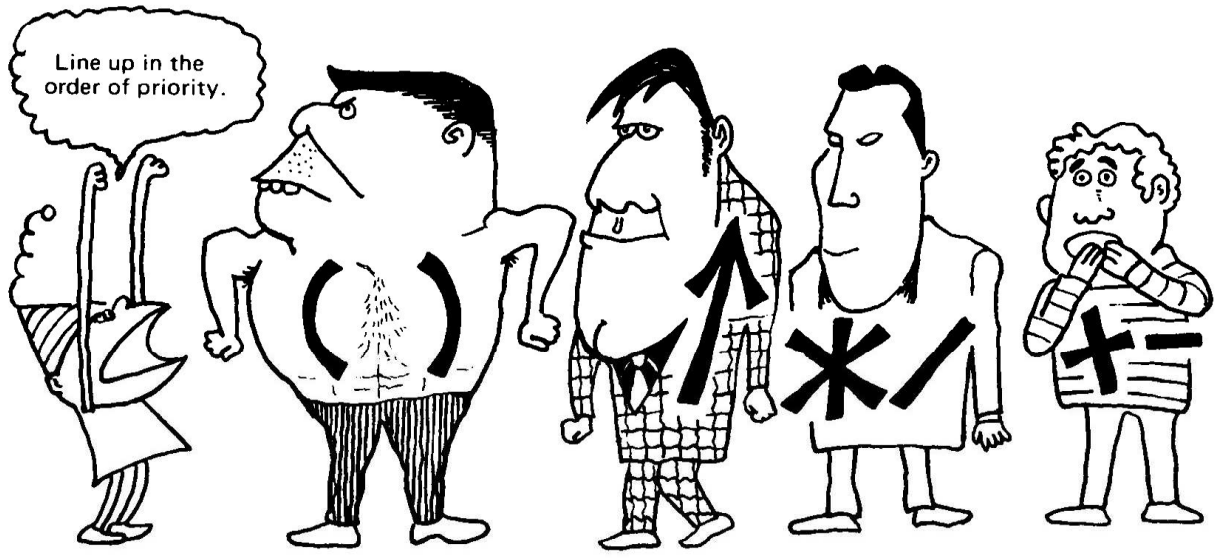
Computing the Earth

The prince of a star takes accurate observation of the earth. "The earth is a blue planet over there in the Solar System. Though slightly distorted, the earth is approximately 13,000 kilometers in diameter. From orbit calculation, its mass is about 6×10^{18} thousand tons."

The prince went to his computer to generate the following program for calculations of volume VE, surface area SE and mean density ZE of the earth.

```
10 DE = 13000          ← Substitute the earth's diameter for variable DE.
20 WE = 6E + 18        ← Substitute the earth's mass for variable WE.
30 SE = 4 * π * (DE/2)↑2          ← This substitutes the surface area for variable SE.
40 VE = 4 * π * (DE/2)↑3/3        ← This substitutes the earth's volume for variable VE.
50 ZE = WE/VE * (1E-2)            ← This substitutes the mean density for variable ZE.
60 PRINT "EARTH DIAMETER" : PRINT DE;" KILOMETERS" : PRINT
70 PRINT "EARTH SURFACE AREA" : PRINT SE;" SQUARE KILOMETER" : PRINT
80 PRINT "EARTH VOLUME" : PRINT VE;" CUBIC KILOMETER" : PRINT
90 PRINT "EARTH MASS" : PRINT WE;" THOUSAND TONS" : PRINT
100 PRINT "EARTH MEAN DENSITY" : PRINT ZE;" KILOGRAM/CUBIC METER"
110 END
```

The prince of a start noticed that the size of the earth has slightly changed. Pay much attention to the units used in the calculations. Further attention is focused on the sequence of calculations when the arithmetic expression contains, * , + or ↑. The operation priority is shown below:



The expressions below are complex in combination. Do you see any difference between the expressions?

```
2 + 3 ↑ 2 = 11
(2 + 3) ↑ 2 = 25

12/3 * 2 = 8
12/(3 * 2) = 2
```

Archimedes and the Mysterious Soldier

The sum of the interior angles of a triangle is 180°. With a flash of inspiration, Archimedes sat on the road and drew a triangle. There came a mysterious soldier with his spear pointing at Archimedes.

Soldier: Archinedes,your life is finished. Be prepared to die !
Archimedes: Wait a minute, I will finish this calculation.

Soldier: What? Angle A is 30° and angle B is a right angle.
It's easy to determine angle C. 60°. If side CA length is known, side AB and BC lengths or even the area of the triangle can be easily determined.
Archimedes: Don't be silly.
Soldier: All that needed is to generate a BASIC program. Let me see, Oh, Yes, it's good with CA = 12.

```
10 A = 30 : B = 90 : CA = 12
20 AB = CA * COS (A * π/180)
30 BC = CA * SIN (A * π/180)
40 S = AB * BC/2
50 C = 180 - A - B
60 PRINT "AB = " ; AB, "BC = " ; BC, "CA = " ; CA
70 PRINT "AREA S = " ; S
80 PRINT "A = " ; A, "B = " ; B, "C = " ; C
90 END
```

Using the inverse tangent ATN, let's determine the size of angle C from the side AB and BC lengths known. This requires the following to be keyed-in.

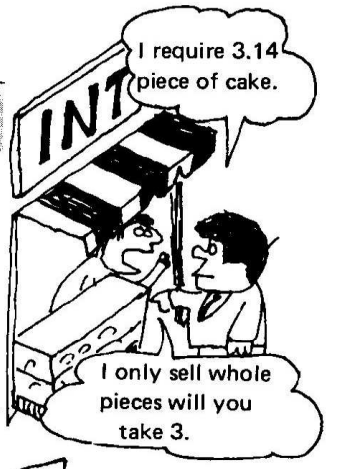
```
50 C = ATN (AB/BC) * 180/π
```

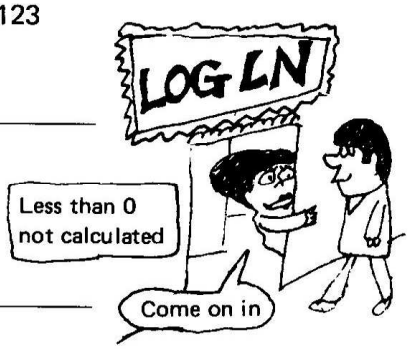
The result is in the unit of degree. The same result is obtained, isn't it?

The Function Family Members

Introduced here are more functions, such as SIN (X). Such functions are used with parentheses, in which constants, variables or arithmetic expressions can be placed.

Function	BASIC Symbol	Calculated Value	Example
Integer	**INT (X)**	Maximum integer within X	INT (3.14) = 3 INT (0.55) = 0 INT (−7.9) = −8
Absolute value	**ABS (X)**	Absolute value of X	ABS (2.9) = 2.9 ABS (−5.5) = 5.5
Sign	**SGN (X)**	1 if X is greater than 0. 0 if X is equal to 0. −1 if X is less than 0.	SGN (500) = 1 SGN (0) = 0 SGN (−3.3) = −1
Exponent function	**EXP (X)**	e^X (e=2.7182818)	EXP (1) = 2.7182818 EXP (0) = 1
Common logarithms	**LOG (X)**	$\log_{10}$ X Provided X is greater than 0.	LOG (3) = 0.47712126
Natural logarithms	**LN (X)**	loge X Provided X is greater than 0.	LN (3) = 1.0986123
Roots	**SQR (X)**	$\sqrt{X}$ Provided X is greater than or equal to 0.	SQR (9) = 3 SQR (0) = 0

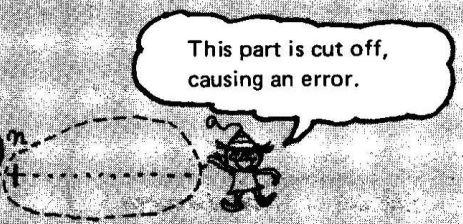




Is PRINT 2*2 Identical to PRINT 2↑2?

Well, 2↑2 results in fractions of 4.0000001. This is correct as an arithmetic expression, but calculations are done in a limited number of figures, involving unexpected errors. For example, 2↑2 is done using the formula called a progression expansion.

$$2\uparrow 2=1+\frac{2\ln 2}{1!}+\frac{(2\ln 2)^2}{2!}+\cdots+\frac{(2\ln 2)^n}{n!}+\cdots\cdots$$



This calculation may cause the computer to scream. The computer will produce certain types of errors. These errors are, however of little concern.

Free Definition of Function ..... DEF FN

Various functions have been described, and here is an explanation of DEF FN defined as a new function combining such various functions. Some definition examples are listed below:

DEF FNA (X) = 2 ✳ X ↑ 2 + 3 ✳ X + 1 . . . $2X^2 + 3X + 1$ is defined as FNA (X).

DEF FNB (X) = SIN (X) ↑ 2 + COS (X) ↑ 2 . . . $\sin^2 X + \cos^2$ is defined as FNB (X) this is always 1.

DEF FNE (V) = 1/2 ✳ M ✳ V ↑ 2 ..... $1/2MV^2$ is defined as FNE (V).

DEF represents "define". New functions are named with FN suffixed. X or V in the parenthesis is called the argument. For example, the third function (seems to be motion energy) is used.

```
10 DEF FNE (V) = 1/2 ✳ M ✳ V ↑ 2
20 M = 5.5 : V = 3.5
30 PRINT FNE (V), FNE (V ✳ 2), FNE (V ✳ 3)
40 END
```

Motion energy at intial velocity V and motion energy with velocity doubled or tripled are displayed. DEF FN command is very convenient particularly when the same functions are often used in a long program.

Fall from an altitude of 10,000 meters!
How do you think the velocity and altitude of a fall from an altitude of 10,000 meters changes per second ?
Function FNV (T) in the program is the fall velocity after a lapse of time T, and FNH (T) is the altitude at the same time.
Acceleration of gravity G, atomospheric resistance factor K and altitude H when a fall occurs are assigned by statement number 20.

```
10 ? "◙" : T = 0
20 G = 9.8 : K = 0.15 : H = 10000
30 DEF FNV (T) = G/K✳(1 – EXP (–1✳K✳T))
40 DEF FNH (T) = H – FNV (T)✳T
50 ? "■"
60 PRINT "TIME      "; T : MUSIC "┌ A0"
70 PRINT "VELOCITY" ; FNV (T)
80 PRINT "ALTITUDE" ; FNH (T)
90 T = T + 1 : GOTO 50
```

← Instruction with beep to be explained on page 84.

← This is a shift instruction for the program to be shifted to statement No. 50. See page 40.

This is INPUT, Answer Please

To inform the computer of variables' values, we have so far taken the method where the value is first determined, as follows:

```
10 A = 3
20 B = 5
.........
```

There are several methods available for informing the computer of the values of variables. One of them uses a command called INPUT.

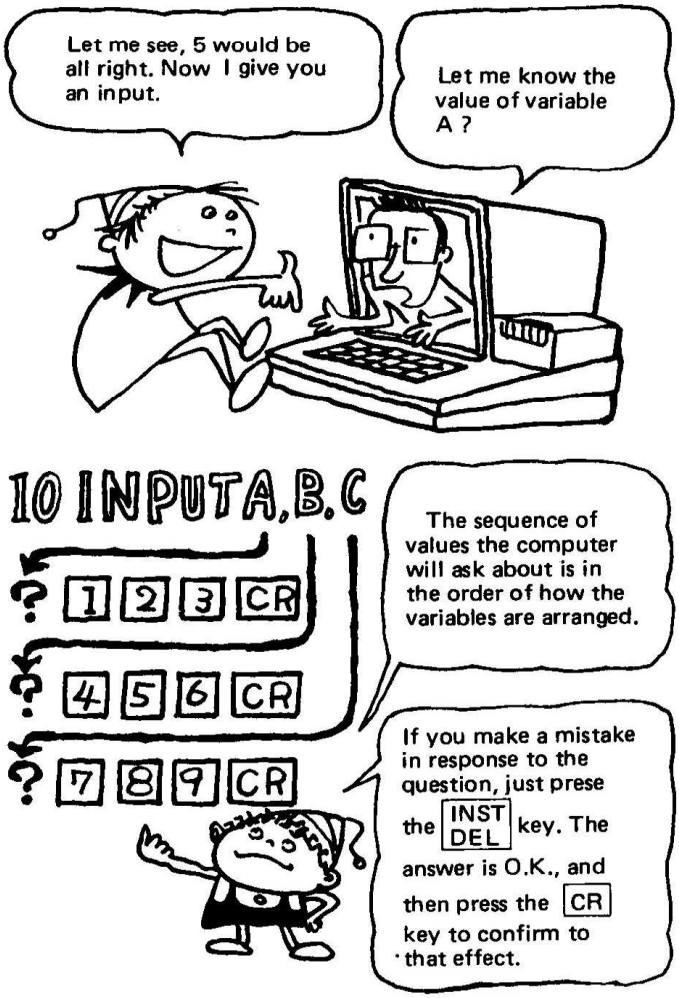
```
10 INPUT A, B, C
20 D = A + B + C
30 PRINT A, B, C, D
40 END
RUN
? ■
```

This is new display, isn't it ? ? ■ is making an inquiry to you about the value of first variable A following the INPUT command. In response to this inquiry, key-in the value and press the CR key to inform the computer that everything is O.K.

Look ! The same display is there. This is the inquiry about the value of the second variable B. If there are 3 variables, the computer asks question 3 times. If you reply using any key other than 0 to 9 by mistake, and press the CR key, the following is displayed.

```
DATA ERROR
```

The computer will then make inquiries about the values all over again.




```
10 INPUT A, B, C, D
20 INPUT E, F, G, H
30 PRINT H, G, F, E
40 PRINT D, C, B, A
50 END
```

STRING

```
10 INPUT "A = ? " ; A
20 INPUT "B = ? " ; B
30 INPUT "C = ? " ; C
40 S = A + B + C
50 M = S/3
60 PRINT "TOTAL" ; S, "MEAN" ; M
70 END
```

SEMICOLON

Yes or No in Reply to a Proposal?

On a sunny Sunday, a gentleman and a lady sit face to face in a nice coffee shop. He is 43 years old, and she is 22 years old.

Gentleman: I love you at first sight. Can you marry me ?
Lady: Yes, if you love me so much. I don't care about the age difference. But not now. You have to wait until my age is half of yours.

Presume his age is A, hers is B and the number of years she asked him to wait is X. After X years, he is A + X years while she is B + X. Since her age is then half of his, the condition of A + X = 2 (B + X) is required. To solve the equation for X, the following is obtained.

X = A − 2 B.

```
10 PRINT "WHAT IS HIS AGE ?"
20 INPUT A
30 PRINT "WHAT IS HER AGE ?"
40 INPUT B
50 X = A − 2 * B
60 PRINT "WAIT" ; X ; " YEARS !"
70 END
RUN
WHAT IS HIS AGE ?
? 43 [CR]
WHAT IS HER AGE ?
? 22 [CR]
WAIT − 1 YEARS !
```

It is impossible to wait for −1 year. In other words, they could have been married a year ago. Asked suddenly about a question, the computer may be confused at what variable you are talking about. In this program, a string indicating inquiry contents is inserted in statement numbers 10 and 30. The string for an answer is also used in statement number 60.
The INPUT method in this program can be simplified. Modify statement numbers 10 and 30 as described below, deleting statement numbers 20 and 40 from the program.

```
10 INPUT "WHAT IS HIS AGE ? " ; A
30 INPUT "WHAT IS HER AGE ? " ; B
```

Semicolon

Those that follow statement number 50 are identical to the above program.

```
RUN
WHAT IS HIS AGE ? 43 [CR]
WHAT IS HER AGE ? 22 [CR]
WAIT − 1 YEARS !
```

DATA and READ go hand in hand

Another method to inform the computer of variables.

```
10 READ A, B, C, D
20 X = A + B + C + D
30 PRINT X
40 DATA 3, 5, 7, 9
RUN
 24
```

This program picks up values which are then used for calculation.
Two types of commands, READ and DATA, are used in this method.

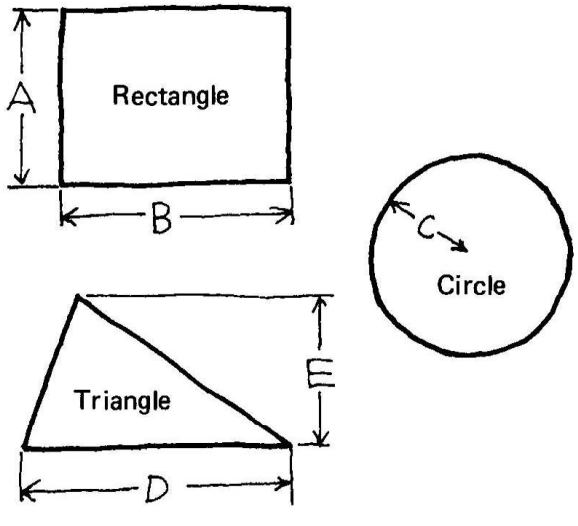
```
READ A, B, C, D ..... Some variables are arranged.
DATA 10, 11, 12, 13 ..... Number of values identical to that of variables that follow READ.
```

Similar to the INPUT command, the arrangements of variables and values must be matched.

READ A, B, C, D
DATA 35, 6, −8, 7 + 5

It is unexpectedly easy to generate programs to determine rectangular, circle and triangle areas using the READ and DATA commands.

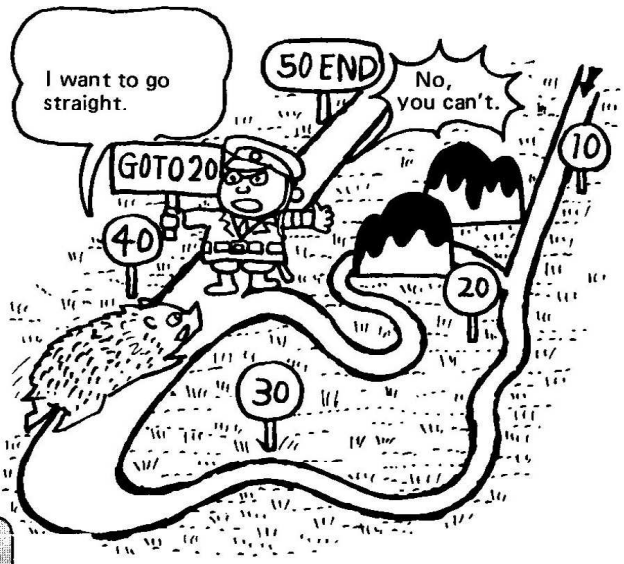
```
30 READ A, B
40 S1 = A * B
50 PRINT "RECTANGLE = "; S1
60 READ C
70 S2 = π * C ↑ 2
80 PRINT "CIRCLE = "; S2
90 READ D, E
100 S3 = D * E / 2
110 PRINT "TRIANGLE = "; S3
120 DATA 8, 10, 5, 8, 10
130 END
```



There seems to be room for improvements in the program.
Try various ways yourself.

Don't Oppose GOTO

For programs described so far, the computer executes them in the correct sequence from small to large statement numbers. In fact, however, execution requires the sequence to be changed on some occasions. On such occasions, GOTO statements are very effective.




```
10 N = 1
20 PRINT N
30 N = N + 1
40 GOTO 20
50 END
RUN
 1
 2
 3
 .
 .
 .
```

GOTO means that an unconditional branch is made to the statement number specified.

Not stopped ? . . . . . Press the [SHIFT] key, then [BREAK] key to stop.

One upon a time, the great Knight Sir Lancelot of the Lake did a great deed for King Arthur of Camelot. King Arthur was so greatful to Sir Lancelot he said, "I would like to give you any prize you care to ask for".
Sir Lancelot replied "Thank you my Lord, I would like to have I Ginea today, 2 Gineas tomorrow, 4 Gineas on the 3rd day, 8 Gineas on the 4th day and so on until the 30th day".
King Arthur was so surprised by such a small request that he agreed immediatly.
Let us make the program below to find out how much King Arthur must pay.

```
10 D = 1 : F = 1 : S = 1
20 PRINT "DAYS", "DAY TOTAL", "TOTAL"
30 PRINT D, F, S
40 D = D + 1 ..... This is for adding oneday to each day.
50 F = 2 * F ..... This is for multiplying oneday's total by two.
60 S = S + F ..... This shows the total by adding to previous day total
70 IF D = 31 THEN 90 .. See page 40.
80 GOTO 30
90 END
RUN
```

NOTE

DAYS	DAY TOTAL	TOTAL
1	1	1
2	2	3
:	:	:
10	512	1023
:	:	:
20	524288	1048575
:	:	:
30	.53687091E + 09	.10737418E + 10

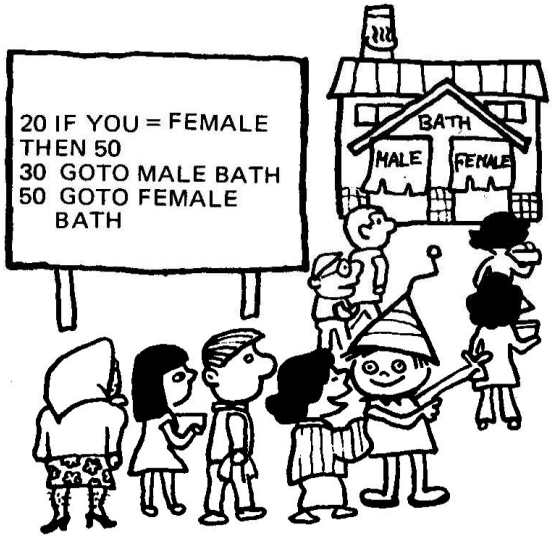
On the 10th day he was given 1023 Gineas, on the 20th day he was given 1048575 on the 30th day he asked for about 1000000000 Gineas.

World of IF.....THEN

Here's the advent of a command that uses the computer more like a computer.

10 IF △△△ THEN □□□
20 ■■■

If △ △ △ conditions are satisfied, then □ □ □ jobs can be executed. If not, omit □ □ □ and go to ■ ■ ■ of the next statement number. This is the IF ~ THEN statement. If □ □ □ is a numberal, a jump is made to the statement number of the numberal.




```
10 READ A
20 IF A >=0 THEN PRINT "A = " ; A
30 GOTO 10
40 DATA -10, 20, 5, -9, 8, -6, 5
50 END
RUN
A = 20
A = 5
A = 8
A = 5
```

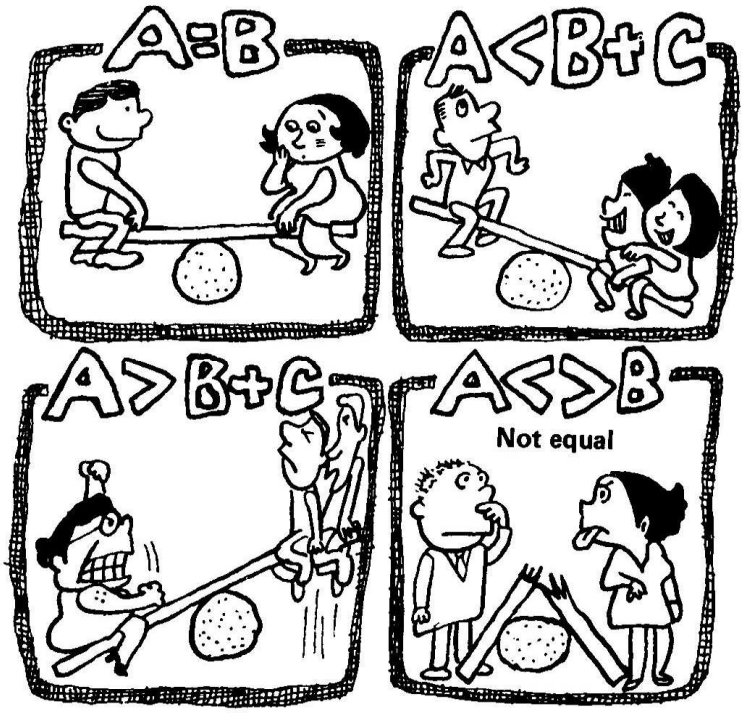
⇐ Positive numbers alone are displayed.

The general form of IF . . . . . THEN statements is as follows:

IF conditions THEN statement or statement number.

The conditions herein referred to are "greater than" or "less than" expressions using equal sign and unequal sign.

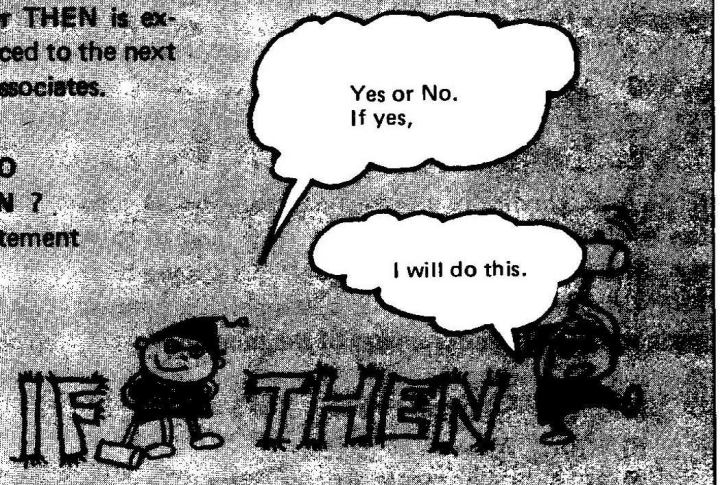
Sign Conditions	How to Use
=	A = B
<	A < B + C
>	A > B + C
< =	A + B < = C
or = <	
> =	A > = B
or = <	
< >	A < > B
or > <	



IF.....THEN and its Associates

If . . . conditions are true (Yes), statement after THEN is executed. If they are false (No), execution is advanced to the next statement number. Here is an introduction to its associates.

IF . . . . . THEN GOTO or IF . . . . . GOTO
IF . . . . . THEN PRINT or IF . . . . . THEN ?
IF . . . . . THEN A = 5 * 7 substitutional statement
IF . . . . . THEN INPUT
IF . . . . . THEN READ
IF . . . . . THEN GOSUB
IF . . . . . THEN RETURN
IF . . . . . THEN STOP
IF . . . . . THEN END




```
10 PRINT "■"
20 PRINT "INSERT OPTIONAL FIGURE FROM 1 TO 9."
25 PRINT "INSERT 0 WHEN YOU STOP."
30 L = 0 : M = 0 : N = 0
40 INPUT A
50 IF A = 0 THEN 90
60 IF A <= 3 THEN L = L + 1 : GOTO 40
70 IF A <= 6 THEN M = M + 1 : GOTO 40
80 N = N + 1 : GOTO 40
90 PRINT "YOU INSERTED FIGURES FROM 1 TO 3" ; L ; " TIMES" ;
100 PRINT " FROM 4 TO 6" ; M ; " TIMES " ;
110 PRINT "AND FROM 7 TO 9" ; N ; " TIMES "
120 END
```

A new symbol ■ is used in statement number 10. Display of ■ is possible when CLR HOME key is pressed, with the SHIFT key depressed, between quotation marks.
This command will clear all the characters on the TV screen and shift the cursor to the top left corner of the TV screen.
In addition, when the CLR HOME key alone is pressed between quotation marks, symbol ■ appears.
This symbol functions only to shift the cursor to the top left corner.
If these are not clear, check with PRINT "■" or PRINT "■".

Leave Any Decision to IF

IF can select Even numbers

Let's consider a program for selecting even numbers only, out of many numerals, using IF . . . .GOTO statement. IF has great ability to select numbers.

```
10 READ X : IF X = –9999 THEN STOP
20 IF X/2 <> INT (X/2) GOTO 10
30 PRINT X ; : GOTO 10
40 DATA 2, 13, 56, 55, 4, 78, 31
50 DATA 6, 22, 15, 19, 80, 11, –9999
RUN
 2 56 4 78 6 22 80
```

INT (X/2) in statement number 20 is the statement for picking integers alone. You remember that, don't you ? Therefore, if X is even, X/2 <> INT (X/2) is impossible, with execution advancing to statement number 30. If it is possible, it's regarded as odd, reading the next value.
To test your progress, let's try an exercise. How can you decide the multiple of 3 or 4 ? You've got it, haven't you ? The answer is this.

Modification for the multiple of three . . . . .

```
20 IF X/3 <> INT (X/3) GOTO 10
```

Modification for the multiple of four . . . . .

```
20 IF X/4 <> INT (X/4) GOTO 10
```

IF can select Maximum and Minimum

```
10 S = 999 : L = –999
20 READ X : IF X = –9999 THEN 80
30 IF X > L THEN L = X
40 IF X > S THEN S = X
50 GOTO 20
60 DATA 2, –5, 91, 256, –43
70 DATA 87, 321, –76, –9999
80 PRINT "MAXIMUM VALUE = " ; L
90 PRINT "MINIMUM VALUE = " ; S
100 END
RUN
MAXIMUM VALUE = 321
MINIMUM VALUE = –76
```

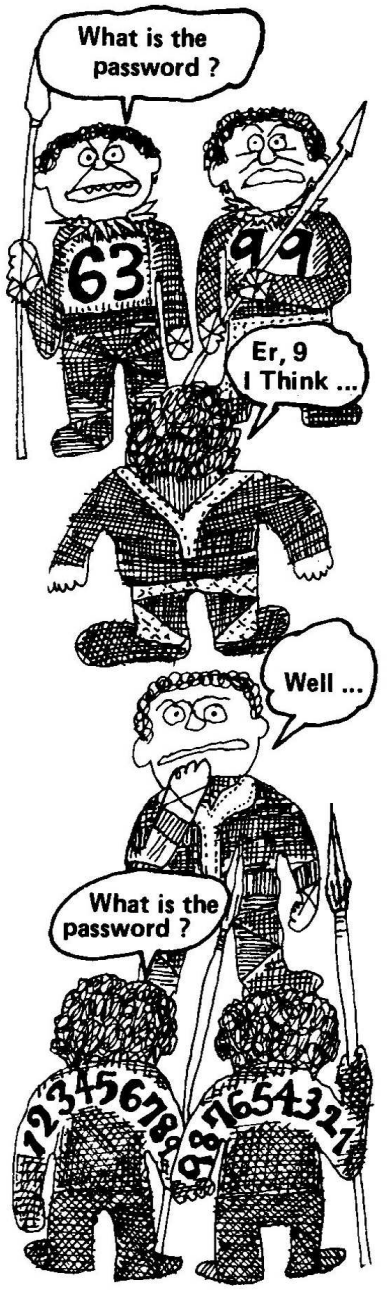
Statement number 10 is very important. Put as large a number as possible in variable S for substitution of the minimum value, and as small a number as possible in variable L for substitution of the maximum value. What about the execution results ? Variable L and S come out as true maximum and minimum values. This is a good example of the use of IF . . . . . THEN.

Password Found for Numbers

The greatest common divisor (GCD) is a password for two integers. For example, presuming that two numbers are 10 and 20, divisible numbers for 10 and 20 are four numbers that are 1, 2, 5 and 10. Of these numbers, the maximum value, namely, 10 is the greatest common divisor for numbers 10 and 20.
Now, let's generate a program.

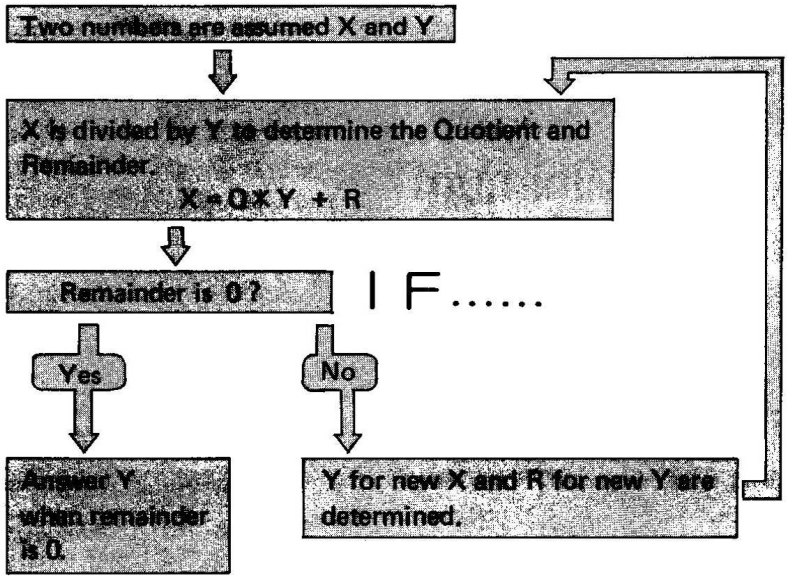
```
10 PRINT "X", "Y", "PASSWORD"
20 READ X, Y
30 PRINT X, Y,
40 Q = INT (X/Y)
50 R = X – Q*Y
60 X = Y : Y = R
70 IF R>0 THEN 40
80 PRINT X : GOTO 20
90 DATA 63, 99, 1221, 121, 64, 658
100 DATA 12345678, 87654321
110 END
 RUN
```

Comma (,) following Y is very convenient for continuous display of data on TV screen.



Exposure of a Trick for this Program !

Long ago, a Greek mathematician, Euclid, developed this method of solution.



Using IF . . . . . , try as many as possible.

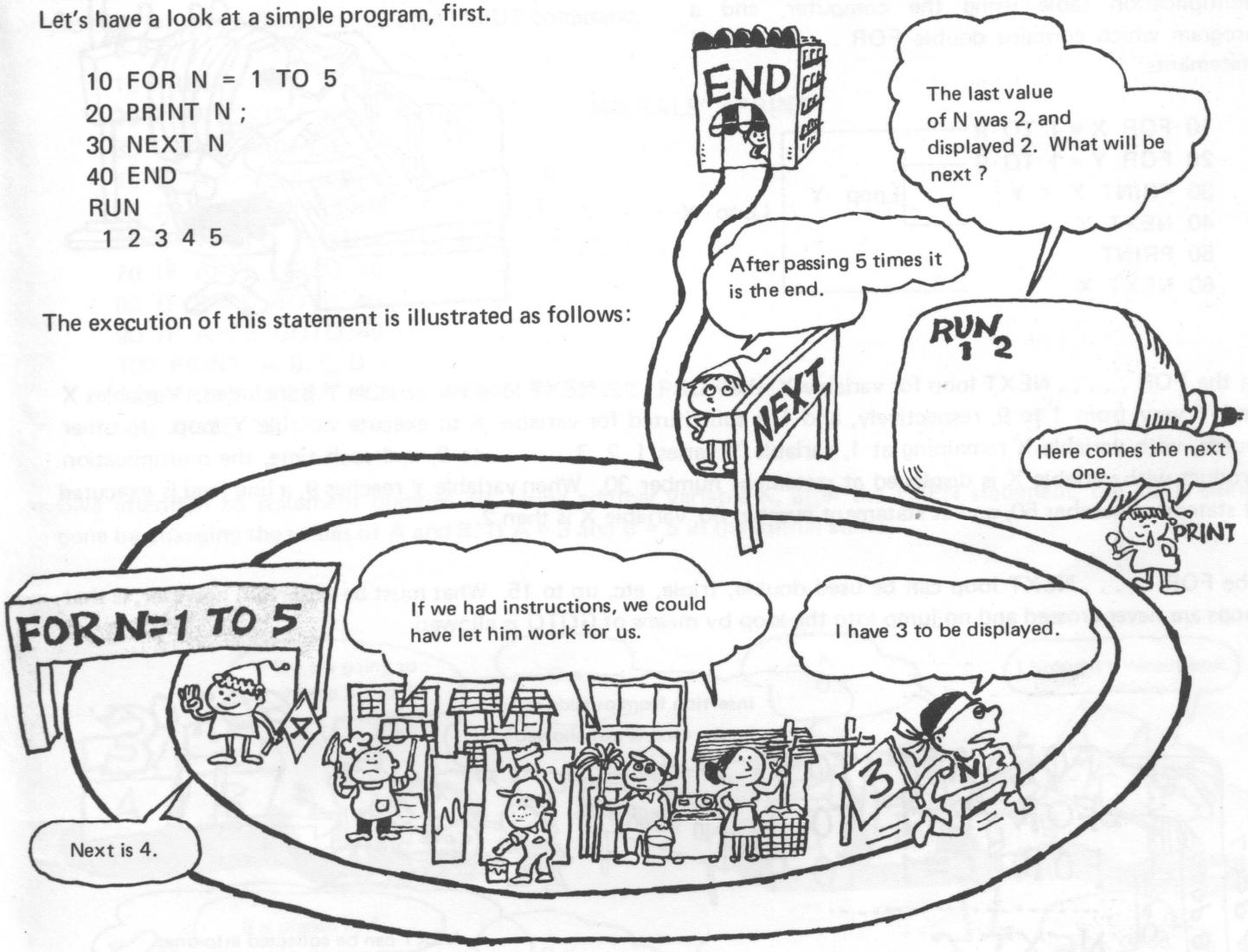
```
10  IF  SGN (X) = –1   THEN ? "MINUS"
20  IF  SGN (X) = 0    THEN ? "ZERO"
30  IF  SGN (X) = 1    THEN ? "PLUS"
```

FOR.....NEXT is an Expert of Repetition

The FOR . . . . NEXT statement is an instruction used for repetition of a sequence of program statements. Let's have a look at a simple program, first.

```
10 FOR N = 1 TO 5
20 PRINT N ;
30 NEXT N
40 END
RUN
 1 2 3 4 5
```

The execution of this statement is illustrated as follows:



The variation of N is not only increased by 1, but can be increased, for example, by 0.5 or decreased by 2. The variation at this time is assigned by the word of STEP.

To increase in 0.5 increments:

```
10 FOR N = 1 TO 5 STEP 0.5
```

To decrease in 2 decrements:

```
10 FOR N = 5 TO 1 STEP -2
```

The general form of FOR . . . . . NEXT statement is as follows:

FOR variable = Initial Value TO Last Value STEP Variation
Repeated Program
NEXT Variable

The initial value, final value and variation may be either a variable, constant or equation.

Loop in a loop

Alice is doing her homework. She is preparing a multiplication table using the computer, and a program which contains double FOR . . . . . NEXT statements.

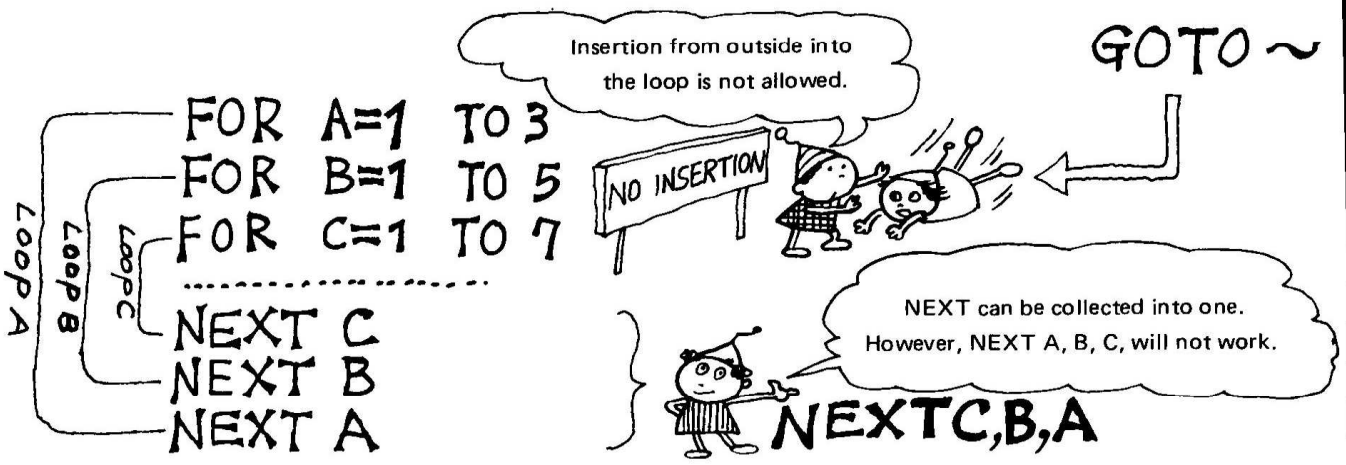

```
10 FOR X = 1 TO 9
20 FOR Y = 1 TO 9
30 PRINT X * Y ;
40 NEXT Y
50 PRINT
60 NEXT X
```

Loop Y

Loop X

In the FOR . . . . . NEXT loop for variable X, the FOR . . . . . NEXT loop for variable Y is included. Variables X and Y vary from 1 to 9, respectively, and 1 is substituted for variable X to execute variable Y loop. In other words, with variable X remaining at 1, variable Y varies 1, 2, 3, ... to 9, and each time, the multiplication product with variable X is displayed at statement number 30. When variable Y reaches 9, a line feed is executed at statement number 50, and at statement number 60, variable X is then 2.

The FOR . . . . . NEXT loop can be used double, triple, etc, up to 15. What must be observed, however, is that loops are never crossed and no jump into the loop by means of GOTO is allowed.



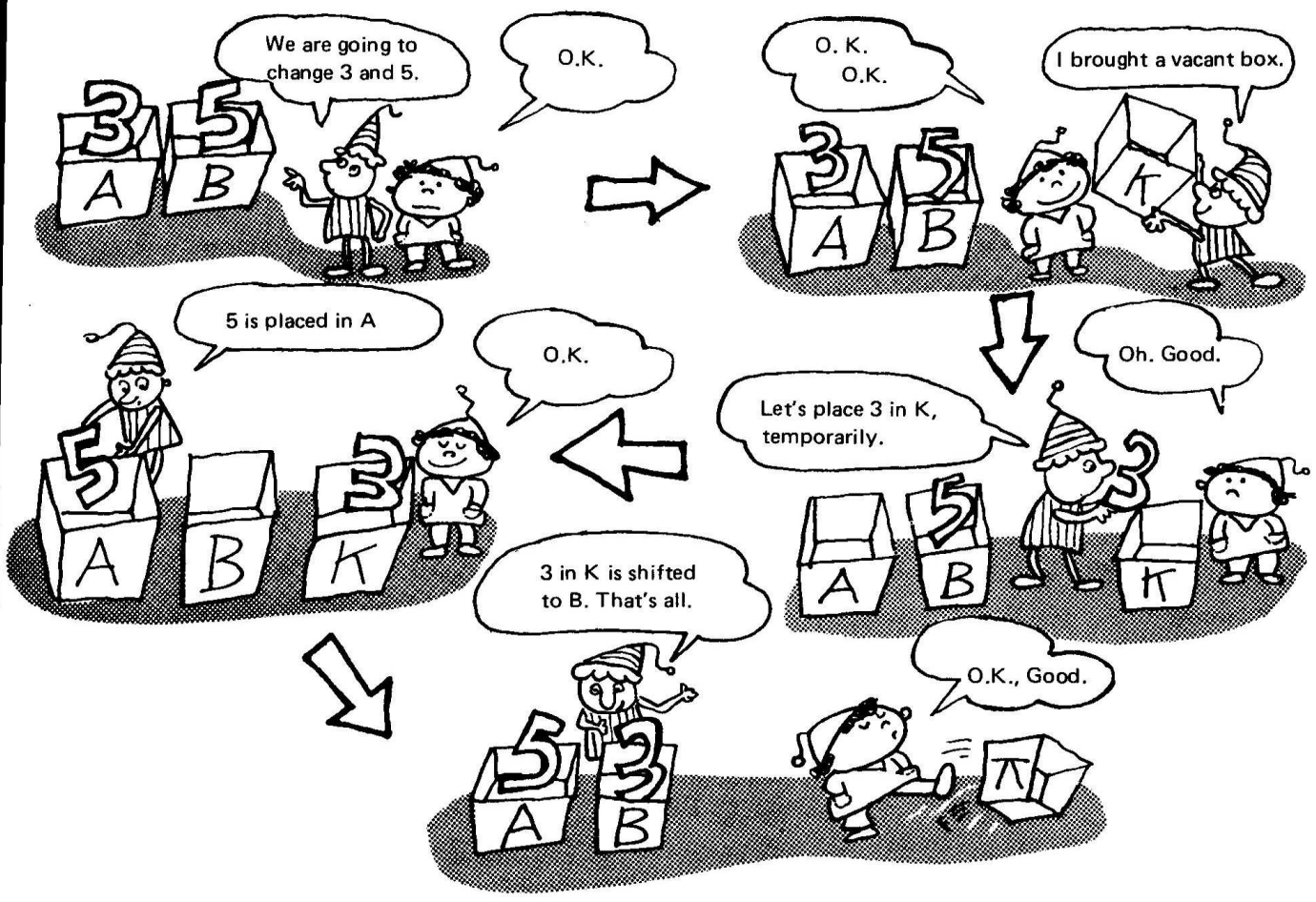
Thus, loop C is completely included in loop B, while loop B is completely included in loop A. As shown on the right, one word NEXT can be used for all three loops.

Line up in Numerical Order

With 4 numericals selected at random and keyed-in, the computer can arrange them in numerical order. This is a program for such a function. Use the INPUT command.

```
10 PRINT "■"
20 PRINT "TELL ME VALUES OF 4 NUMERALS" : PRINT
30 INPUT A, B, C, D
40 IF A<=B THEN K=A : A=B : B=K
50 IF B<=C THEN K=B : B=C : C=K
60 IF C<=D THEN K=C : C=D : D=K
70 IF A<B GOTO 40
80 IF B<C GOTO 40
90 IF A<C GOTO 40
100 PRINT A, B, C, D
110 PRINT : PRINT "ONCE MORE PLEASE" : PRINT
120 GOTO 30
```

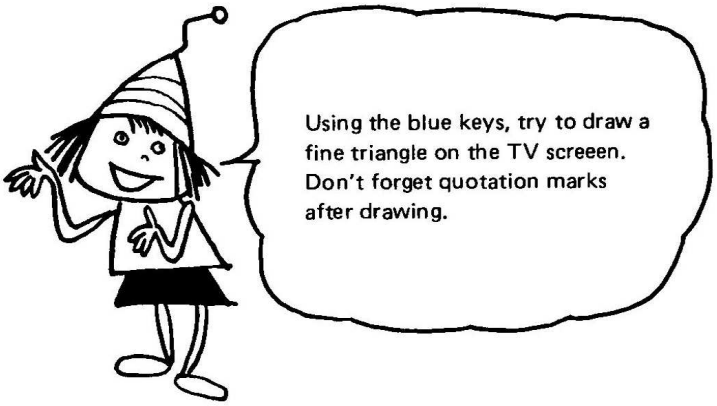
Give attention to statement number 40. Using another variable K, after the THEN statement, the job is being done by changing the values of A and B. If A = 3 and B = 5 in the initnal state;



By the above job, A = 5 and B = 3 are obtained. Similar processing is executed at statment numbers 50 and 60. Statement numbers 70 through 90 are propared for the repetition of the changing job.

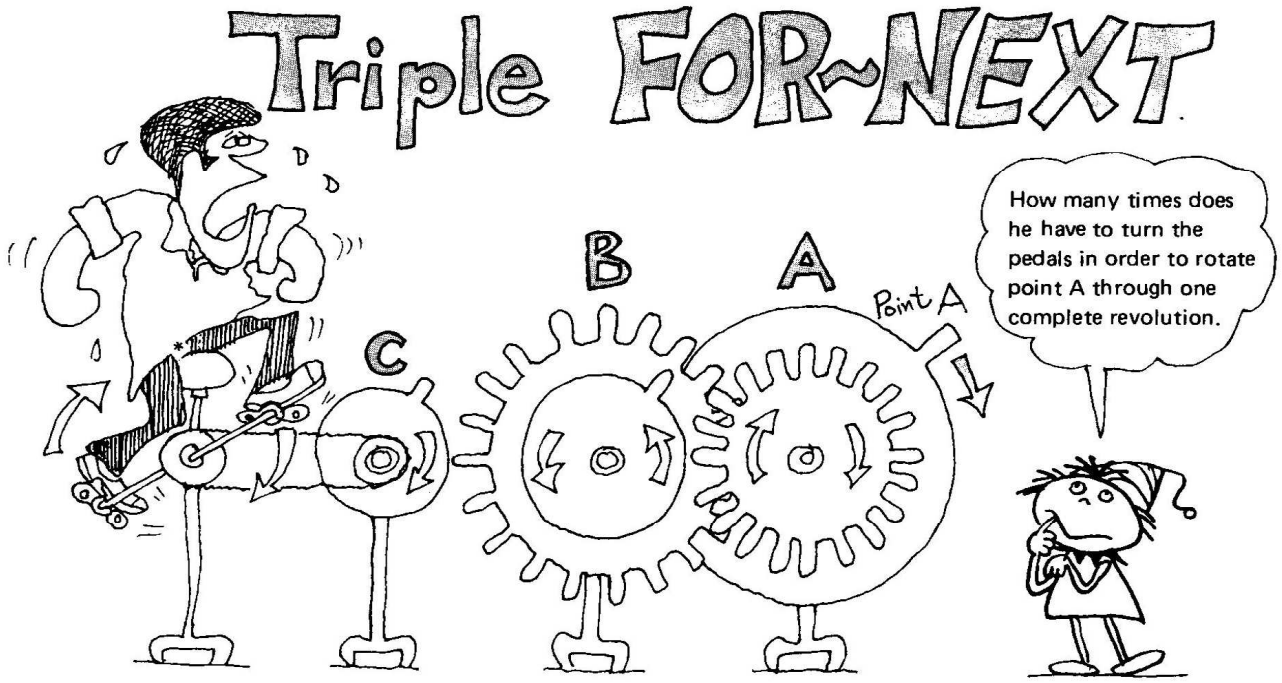
How Many Right Triangles are Possible?

Now, let's generate a program that picks up positive integers from 1 to 20 to meet the Pythagorean theorem $A^2 = B^2 + C^2$.



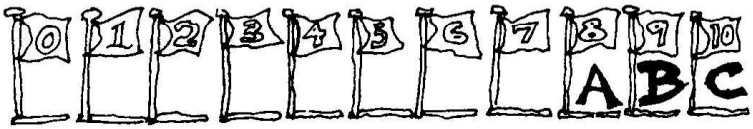

```
10 PRINT "■"
20 PRINT "      |\ "
30 PRINT "      | \ "
40 PRINT "    B |  \ A"
50 PRINT "      |   \ "
60 PRINT "      |    \ "
70 PRINT "      ------ "
80 PRINT "         C"
90 PRINT : PRINT "POSITIVE INTEGERS TO MEET PYTHAGOREAN THEOREM"
110 PRINT : PRINT "➡ A", " ➡B", " ➡ C"
120 FOR A = 1 TO 20
130 FOR B = 1 TO 20
140 FOR C = 1 TO 20
150 IF A * A - B * B - C * C = 0 THEN PRINT A, B, C
160 NEXT C, B, A
180 END
```

You already know the meaning of statement number 10. Try to draw carefully so that a fine triangle is formed between statement numbers 20 through 80. At statement numbers 120 through 160, the FOR . . . . . NEXT loop is triple. The equation shown at statement number 150 is repeated 8000 times (20 x 20 x 20) with C from 1 to 20 at A = 1 and B = 1, and with C from 1 to 20 at A = 1 and B = 2, and so on.
This operation requires a considerable period of time for its completion.

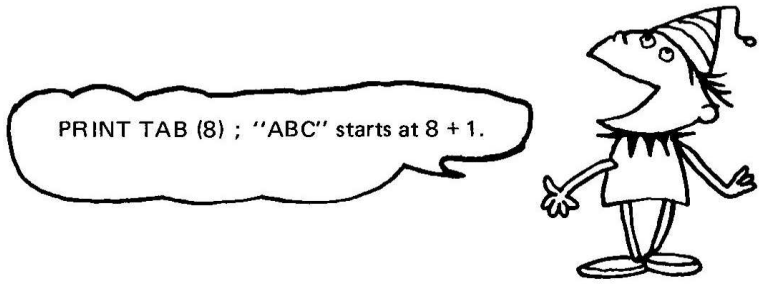


TAB( ) is Versatile

It is possible to assign where to start writing the characters or symbols of a string on the TV screen. The TAB ( ) is used to do so.



Using PRINT TAB (8) ; "ABC", string ABC is displayed at the number in the parenthesis counted from the left hand side, namely, starting at the 9th position.
The numbers to be assigned for the parenthesis are from 0 to 78, and variables may be used if defined as numerals.



Let's operate an example of a simple program combined with the FOR . . . NEXT statements.

```
10 FOR X = 1 TO 20
20 PRINT TAB (X) ; " ✳ "
30 NEXT X
```

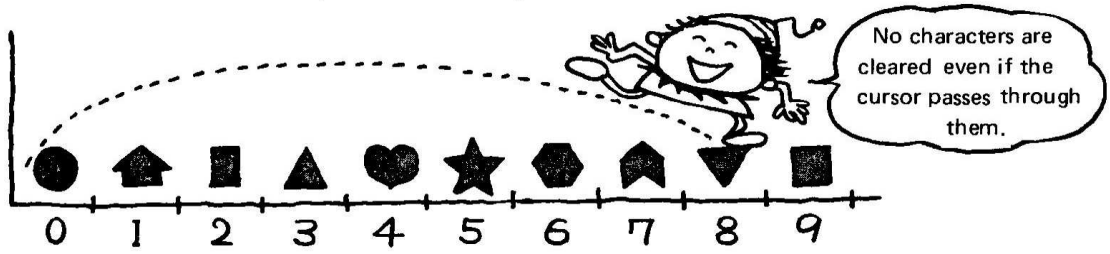
```
10 FOR Y = 1 TO 20
20 PRINT TAB (20 – Y) ; " ✳ "
30 NEXT Y
```

Now, let's try a little more complex program.

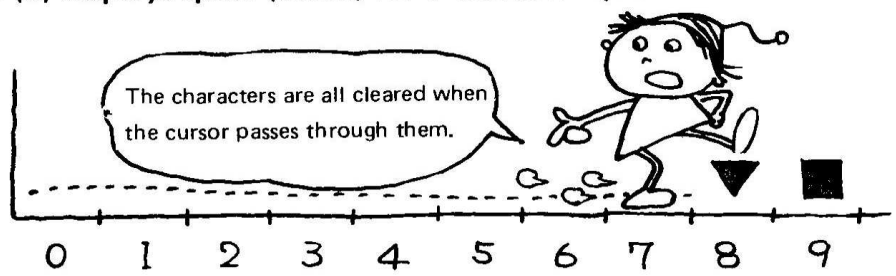
```
10 PRINT " ■ " : PRINT SPC (8) ;
20 FOR X = 1 TO 22 : PRINT " ✳ " ; : NEXT X : PRINT
30 FOR Y = 1 TO 20
40 PRINT TAB (8) ; " ✳ " ; TAB (29 – Y) ; " ■ " ; TAB (29); " ✳ " : NEXT Y : PRINT SPC (8) ;
50 FOR Z = 1 TO 22 : PRINT " ✳ " ; : NEXT Z
```

A new statement is there at statement numbers 10 and 40. When this SPC ( ) is substituted for TAB, exactly same result is obtained. However, there is the difference shown below between the SPC and TAB.

TAB (8) shifts the cursor by 8 character space from the left hand on TV screen.



SPC (8) displays space (blank) for 8 character space.



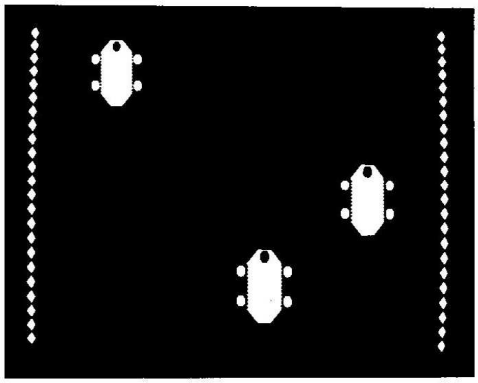
Grand Prix Using RESTORE

Challenge of a Car Race .....

How about the simplest car race program ?

```
10 X = 33 * RND (1)
20 FOR A = 1 TO 5
30 READ M$
40 PRINT TAB (0) ; " ♦ " ; TAB (X) ; M$ ;
50 PRINT TAB (37) ; " ♦ "
60 NEXT A
70 Y = 10 * RND (1)
80 FOR A = 1 TO Y
90 PRINT TAB (0) ; " ♦ " ;
100 PRINT TAB (37) ; " ♦ " : NEXT
110 RESTORE : GOTO 10
120 DATA "  ", "  "
130 DATA "  ", "  "
140 DATA "  "
```

This instruction generates random numbers.
(Refer to page 70.)



TAB (X) at statement number 40 determines where to display automobiles on the road, particularly from the left side. What distance between the automobiles ? For this, random numbers from 1 to 9 are generated at statement number 70, and at statement numbers 80 through 100, automobiles are controlled so that they do not collide. By the way, RESTORE at statement number 110 is not a familiar command, is it ?

RESTORE Returns to the Start of Data

No matter where it may be, or no matter how it may be scattered, DATA statement is read by READ statement.

O. K.

```
10 DATA 27
20 READ A, B, C
30 DATA 10
40 .....
50 DATA 9, 13
60 READ D
70 END
```

NO

```
10 READ A, B
20 READ C
30 DATA 27, 10, 9
40 READ D
50 END
```

Why No ? Because variable D has no value of DATA to be read.
Then what about this ?

```
10 READ A, B
20 READ C
30 DATA 27, 10, 9
35 RESTORE
40 READ D
50 END
```

RESTORE statement enables the reading of the first data in the first DATA statement of the program.

Talkative Strings

The computer should generate a language understandable to human beings and should talk to us . . . . To make such a desire come true, string variables are absolutely necessary.

```
10 A$ = "MIKE" : B$ = "PAUL"
20 C$ = "TONY" : D$ = "PETE"
30 E$ = "DENIS" : F$ = "MARTIN"
40 G$ = "PHILIP"
50 I$ = "JACK" : J$ = "HARRY"
60 K$ = "BILL" : L$ = "DAVID"
```

Ordinary variable symbols with $ (dollar sign) suffixed are called string variables. Processing, very similar to that of ordinary variables is possible. Let's look at some examples to see their characteristics.

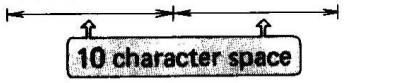
```
70 PRINT B$
80 PRINT A$
RUN
PAUL
MIKE
```

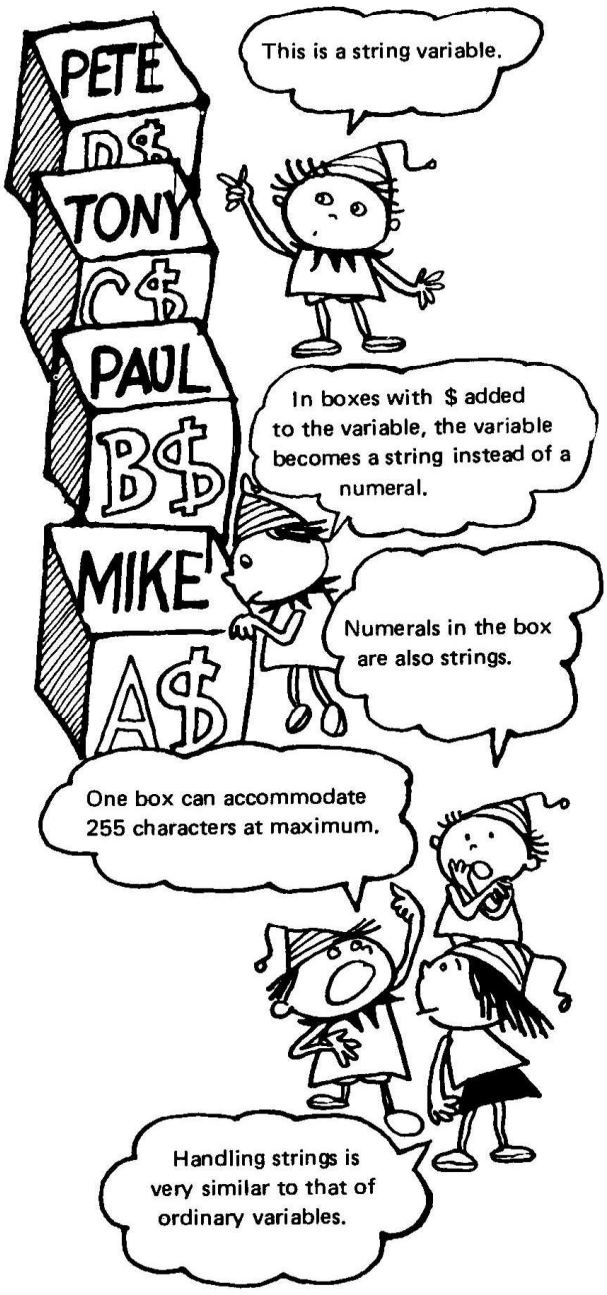
Using " ; ", connects string variables.

```
100 PRINT B$ ; C$ ; A$ ; E$ ; L$ ; D$ ; K$
RUN
PAULTONYMIKEDENISDAVIDPETEBILL
```

What will happen if " , " is used in place of " ; " ?

```
120 PRINT B$, A$, I$
RUN
PAUL          MIKE          JACK
```

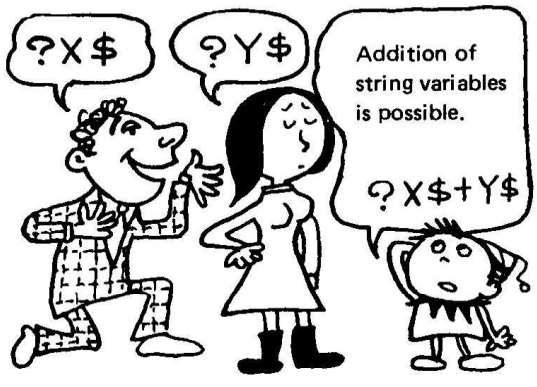




To combine string variables to generate a new string, add string variables together using " + ". Let's try one.

```
140 X$ = B$ + C$ + A$ + D$ + J$ + G$
150 Y$ = A$ + C$ + B$ + F$ + I$ + G$
160 PRINT X$
170 PRINT Y$
```

With this, a new string variable is possible.



Another type of INPUT

Combine string variables and INPUT statement in a program to create a poem.

```
10 INPUT A$, B$, C$
20 PRINT A$ ; " " ; B$ ; " " ; C$
30 GOTO 10
RUN
```

Space for syllable separations.

```
? A FROG JUMPS
? INTO A POND
? WITH A SPLASH OF WATER.
A FROG JUMPS INTO A POND WITH A SPLASH O
F WATER.
```

Using INPUT statement, the input of a string can be keyed-in, requring no quotation marks " "

Another example of this is shown below.

```
10 PRINT "TYPE IN ANYTHING AT ALL"
20 INPUT AA$
30 PRINT "YOU HAVE JUST TYPED " ; AA$
40 GOTO 10
```

String variables, when combined with READ and DATA statements, can be generated into a program.

```
10 READ X1$, X2$
20 PRINT X1$; "LIKE CREAM " ; X2$
30 DATA DO YOU , CAKES ?
RUN
DO YOU LIKE CREAM CAKES ?
```

Note that quotation marks are not required when READ statement is used.

LEFT$, MID$, RIGHT$

LEFT $ ( ), MID $ ( ) and RIGHT $ ( ) are statements to generate new strings by taking out part of strings.

```
10 A$ = "AQUARIUS PISCES ARIES LEO"
20 B$ = LEFT$ (A$, 15)
30 PRINT B$
RUN
AQUARIUS PISCES
```

Character up to the 15th from the left hand side

LEFT $ (A$, 15) selects the characters up to the 15th out of the string A$ in order to generate a new string. The string variable B$ has been defined for the new string.

To select some characters when counted from the right hand side of a string, RIGHT $ ( ) is used.

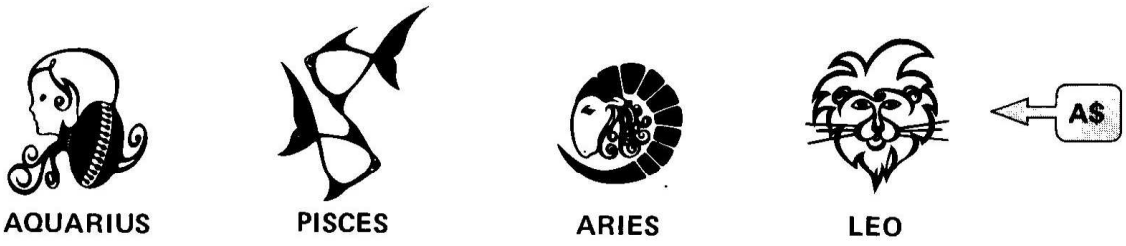
```
40 C$ = RIGHT $ (A$, 9)
50 PRINT C$
RUN
ARIES LEO
```

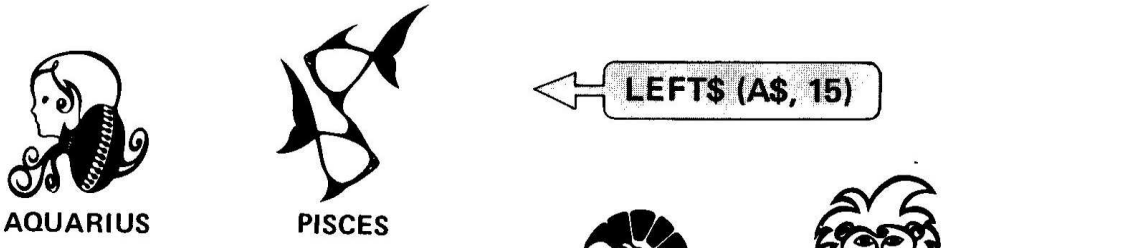
Selects the last 9 characters from A$

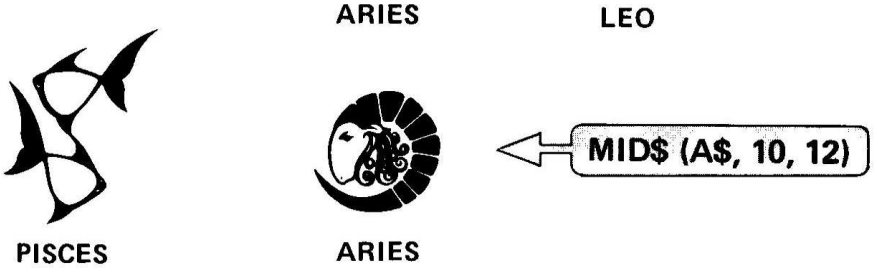
To select some characters in the centre of a string, MID$ ( ) is used.

```
60 D$ = MID$ (A$, 10, 12)
70 PRINT D$
RUN
PISCES ARIES
```

Selects 12 characters starting at position 10 of A$







LEN is a Measurement for Strings

LEN ( ) is used to discover the character count of a string. A simple example of this statement is as follows:

```
10 A$ = "ABCDEFG"
20 PRINT LEN (A$)
RUN
 7
```

The character count of a string variable A$, namely "7" is displayed.
Here is a program using LEN ( ) statement for drawing a square.

```
10 PRINT "■" : PRINT " TYPE HORIZONTAL SIDE USING * KEY"
20 INPUT A$
30 FOR J = 1 TO LEN (A$) −2
40 PRINT TAB (2) ;" * " ; SPC (LEN (A$) − 2) ;" * "
50 NEXT J
60 PRINT TAB (2) ; A$ : GOTO 20
```

Vary the values of * INPUT. The computer performs square drawing by using LEN ( ). Then, drawing is made possible by characters or symbols other than " * ". Using LEFT $ ( ), statement numbers 20 and 40 of the previous program are modified.

```
20 INPUT A$ : AA$ = LEFT $ (A$, 1)
40 PRINT TAB (2) ; AA$ ; SPC (LEN (A$) − 2) ; AA$
```

The use of LEN makes a string parade possible.

```
10 S$ = "SHARP BASIC"
20 FOR M = 1 TO LEN (S$)
30 PRINT LEFT$ (S$, M)
40 NEXT M
RUN
S
SH
SHA
SHAR
SHARP
SHARP
SHARP B
SHARP BA
SHARP BAS
SHARP BASI
SHARP BASIC
```

```
10 S$ = "SHARP BASIC"
20 FOR M = 1 TO LEN (S$)
30 PRINT RIGHT$ (S$, M)
40 NEXT M
RUN
C
IC
SIC
ASIC
BASIC
 BASIC
P BASIC
RP BASIC
ARP BASIC
HARP BASIC
SHARP BASIC
```

ASC and CHR$ are Relatives

ASC

```
10 PRINT ASC (" A ");
20 PRINT ASC (" ABC ");
30 T$ = "Z" : PRINT ASC (T$)
40 END
RUN
 65 65 90
READY
```

With strings in the parenthesis ( ) of ASC, when PRINT is keyed-in the result always shows numerals. Actually, this shows the ASCII code. All characters used with the computer are based on the ASCII code. For its table, refer to page 121. ASC ( ) picks up the ASCII code for the first character of string in the parentnesis ( ). This gives a clear clue to the reason why the same result is obtained although the strings in the parentheses differ between statement numbers 10 and 20. The ASCII code is for characters up to 255.

CHR$

If characters can be converted to the ASCII code, it is natural that there is a statement to reverse the conversion. That's right. CHR$ statement does that job.

```
PRINT CHR$ (65), CHR$ (ASC (" K "))
A             K
READY
```

A cipher is generated using the numerals. Let CHR$ read it.

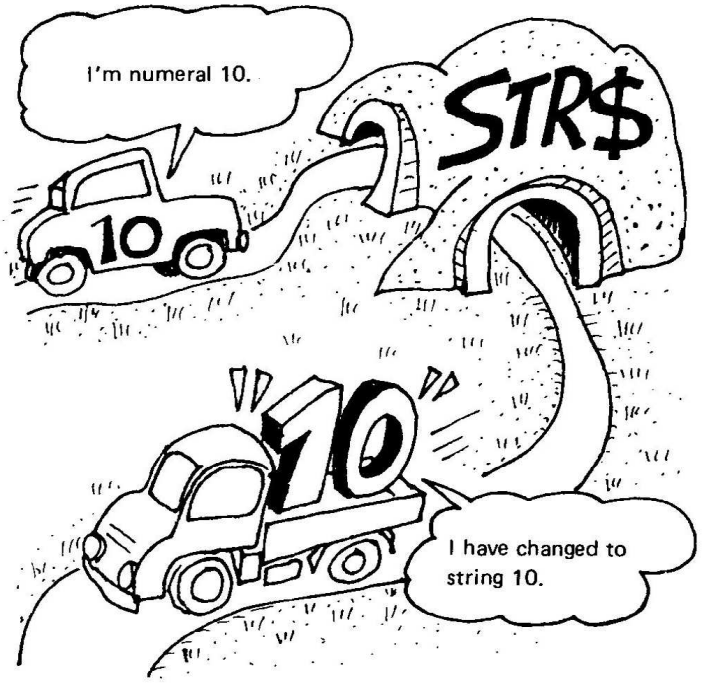
```
10 FOR J = 1 TO 24 : READ A
20 B$ = CHR$ (A)
30 PRINT B$ ; : NEXT : END
40 DATA 73, 196, 83, 84, 85, 68
50 DATA 89, 196, 66, 65, 83, 73
60 DATA 67, 196, 79, 70, 196, 77
70 DATA 90, 45, 56, 48, 75, 46
RUN
I – STUDY – BASIC – OF – MZ – 80K.
READY
```

STR$ and VAL are Numeral Converters

STR$

```
10 A = 12 : B = 3 : C = A + B
20 C$ = STR$ (A) + STR$ (B)
30 PRINT C, C$
40 END
RUN
 15          123
READY
```

The value of variable A is converted to a string of characters by STR$ (A) and string-processed. The reason why C$ contents are 123 is clear to you. In the following program, use STR$ to match the " . " of data.



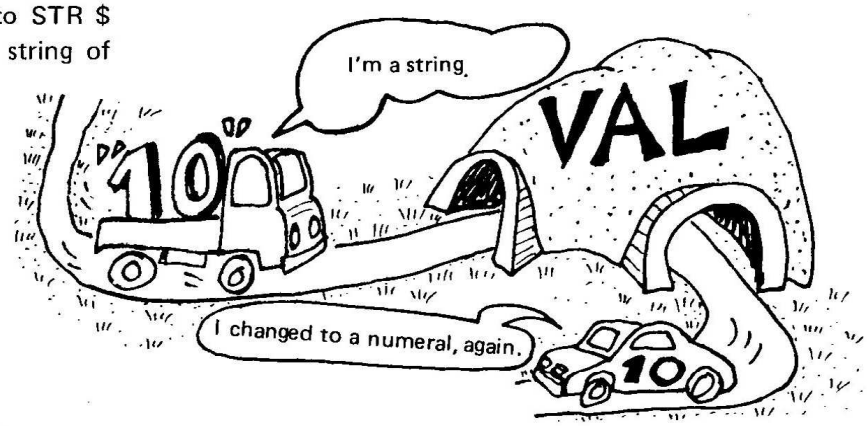

```
10 FOR X = 1 TO 5
20 READ A
30 L = 5 – LEN (STR$ (INT (A) ))
40 PRINT TAB (L) ; A
50 NEXT : END
60 DATA 1.23456, 12.3456
70 DATA 123.456, 1234.56
80 DATA 12345.6
```

The results of the program on the left are as follows.

```
    1. 23456
   12. 3456
  123. 456
 1234. 56
12345. 6
READY
```

VAL

VAL statement has a function opposite to STR $ statement. In other words, it converts a string of characters to a numeral.

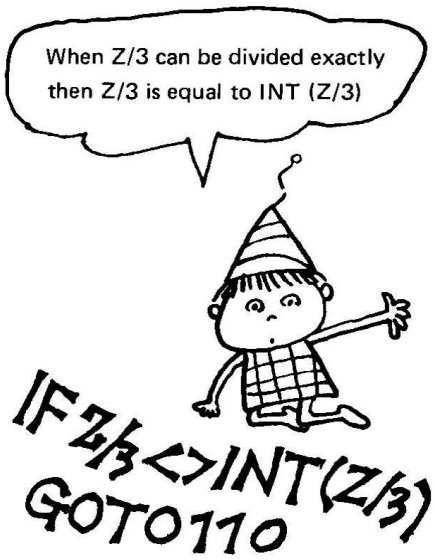



```
10 A$ = " 123456 "
20 B = VAL (A$)
30 C = 654321 + B
40 PRINT A$
50 PRINT B
60 PRINT C
80 END
RUN
123456          . . . . . .  Not a numeral but a string.
 123456         }. . . . . .  Numeral, so there is a space for ± (plus/minus sign) to be placed
 777777                       before the most significant digit of the numeral. For a negative
READY                         numeral, a minus sign is placed in the space.
```

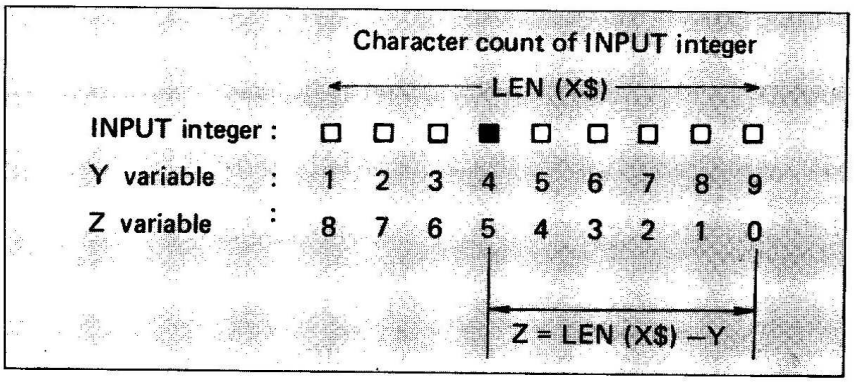
Print out as £123,456,789 .....

This program reads an integer of an optional figure under the INPUT statement, and writes it adding commas (,) to every 3 figures from the right. Given 0 as an integer, the program terminates.

```
10 PRINT " INPUT INTEGER ";
20 INPUT X$
30 IF X$ = " 0 " THEN END
40 PRINT " £ " ;
50 FOR Y = 1 TO LEN (X$)
60 PRINT MID$ (X$, Y, 1) ;
70 Z = LEN (X$) – Y
80 IF Z/3 <> INT (Z/3) GOTO 110
90 IF Z = 0 GOTO 110
100 PRINT " , " ;
110 NEXT Y
120 PRINT : PRINT : GOTO 10
RUN
INPUT INTEGER ? 1 2 3 4 5 6 7 8 9
£ 123,456,789

INPUT INTEGER ? 1 2 3 4
£ 1,234
```



Stament number 80 checks to see if Z (Character position counted from the right) is a multiple of 3. If so, a comma " , " is placed at statement number 100. For example, presuming that the input integer is a number of 9 figures, the following is obtained.



Take a number consisting of figures 1 to 4, and another number of the same figures but with a reverse arrangement to the former, then add up these two numbers. You will thus find that the sum is the same whether it is counted from the right or from the left.

```
10 PRINT "■ENTER SOME NUMBER COMPOSED OF FIGURES 1 TO 4 (WITHIN 8 DIGITS)"
20 Z$ = "" : INPUT X$
30 FOR K = LEN (X$) TO 1 STEP –1
40 Y$ = MID$ (X$, K, 1)
50 Z$ = Z$ + Y$ : NEXT K : X = VAL (X$) + VAL (Z$)
60 PRINT X : PRINT : GOTO 20
```

What's the Difference between the Simple and Compound Interests?

£ 10,000 is deposited in a bank, and one year later, £ 10,600 is drawn. Interest rate, in this case, is found to be 600/1000 = 0.06 or 6%. Then, what is the interest when deposited for 2 years. There are two methods available for interest calculation. One is simple interest calculation based on the fact that the interest of £ 600 for the first year doubles for the second year, amounting to £1200. The other is compound interest calculation based on the idea that a deposit at the beginning of the second year is £ 10,600 with interest of £ 636 (£10,600 x 0.06) added to make £ 1236 for two years. Compound interest calaculation is slightly better in interest rate. For a larger sum of money deposited for longer terms, the difference in interest rate between the two methods must be noticeable. The following is the equation for determining interest included for the n year in each calculation method.

Interest included by simple calculation (n year with rate R)

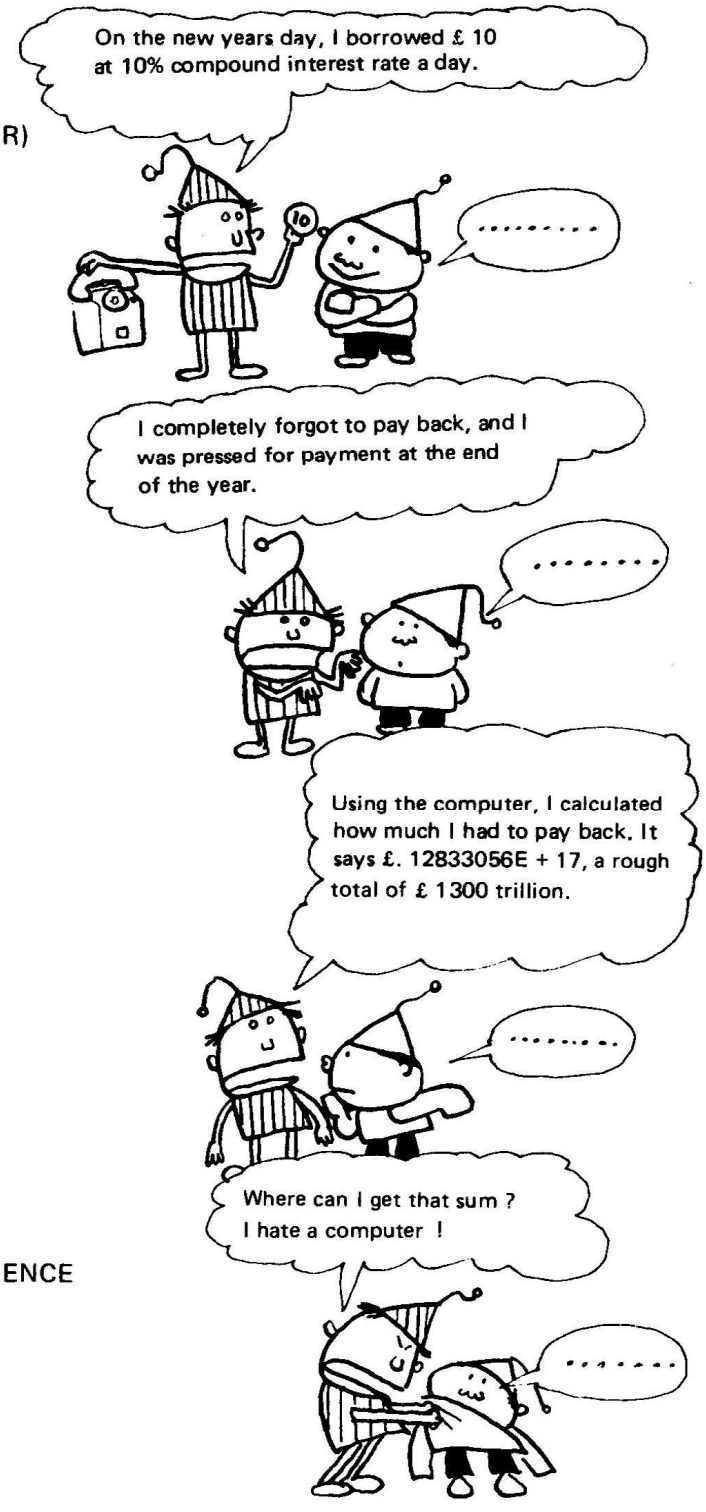
$$B = X \text{ (principal)} + n \cdot X \cdot R$$

Interest included by compound calculation (n year with rate R)

$$C = X \cdot (1 + R)^n$$

Based on the above equations, the following program is generated to calculate interest included both in simple and compound interests.

```
10 PRINT "PRINCIPAL"
20 INPUT X
30 PRINT "INTEREST RATE %"
40 INPUT R
50 PRINT "NUMBER OF YEARS"
60 INPUT Y : PRINT : PRINT
70 PRINT "PRINCIPAL = " ; X
80 PRINT "INTEREST RATE = " ; R ; " % "
90 PRINT "YEARS" ; TAB (6) ; "SIMPLE" ;
100 PRINT TAB (17) ; "COMPOUND";
110 PRINT TAB (30) ; "DIFFERENCE"
120 FOR A = 1 TO Y
130 B = X + A * X * (R/100)
140 C = INT (10 * X * (1 + R/100) ↑ A)/10
150 D = C – B
160 PRINT A ; TAB (6) ; B ;
170 PRINT TAB (15) ; C ; TAB (30) ; D
180 NEXT A
190 PRINT : PRINT : GOTO 10
```



The following is an example of program execution:

```
PRINCIPAL = 10000
INTEREST RATE = 6%
```

YEARS	SIMPLE	COMPOUND	DIFFERENCE
1	10600	10600	0
2	11200	11236	36
3	11800	11910.1	110.1
4	12400	12624.7	224.7
5	13000	13382.2	382.2
6	13600	14185.1	585.1
7	14200	15036.3	836.29999

Annuity if Deposited for 5 Years

In the previous example, we looked at the difference in interest between the simple and compound interest calculations for money deposited. Actually, however, monthly deposit, like fixed deposit, is more familiar to us. If a fixed amount of money X is deposited monthly, the interest included increases with X (1 + R) for the first year, X $(1 + R)^2$ for the second and so on. In addition, when sum X is deposited yearly, the money to be deposited the year after, 2 years from now, will be X (1 + R). Such an increase of deposits is shown below in equations:

Interest included a year after (Principal X and interest R)

$$M_1 = X\ (1 + R)$$

Interest included 2 years after

$$M_2 = X\ (1 + R)^2 + X\ (1 + R)$$

Interest included 3 years after

$$M_3 = X\ (1 + R)^3 + X\ (1 + R)^2 + X\ (1 + R)$$

Based on the above, the interest included is calculated in the following equation for n years.

$$M_n = X\ (1 + R)^n + X\ (1 + R)^{n-1} \ldots\ldots + X\ (1 + R)$$

This is simplified as follows:

$$M_n = X\ ((1 + R)^{n+1} - (1 + R))/R$$

Here's the program generated to indicate what is the interest included for any desired year with the same amount of money deposited each year. Even though the same amount is deposited, this program is designed to allow inputs of minimum and maximum amounts.

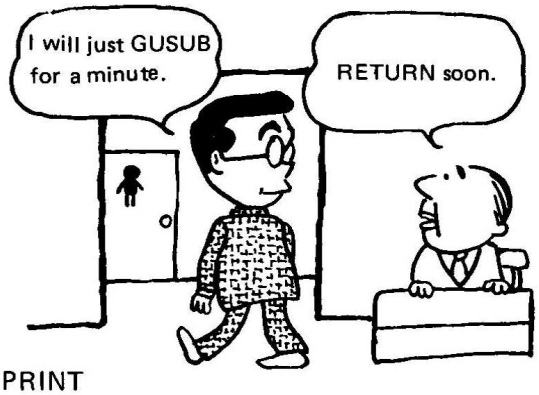
```
10 PRINT "INTEREST RATE %" ;
20 INPUT R
30 PRINT "ENTER AMOUNTS"
40 PRINT "MINIMUM" ; : INPUT L
50 PRINT "MAXIMUM" ; : INPUT H
60 PRINT "NUMBER OF YEARS" ;
70 INPUT Y : PRINT : PRINT
80 PRINT "RATE" ; R ; " % "
90 PRINT "EACH YEAR" ; TAB (12) ; Y ; "YEARS"
100 R = R/100 : PRINT
110 FOR A = L TO H STEP 10000
120 B = INT (A * ((1 + R) ↑ (Y + 1) - (1 + R)) /R)
130 PRINT A ; TAB (12) ; B
140 NEXT A
150 PRINT : PRINT : GOTO 10
```

The Result of Program Execution

```
INTEREST RATE %? 10
ENTER AMOUNTS
MINIMUM? 50000
MAXIMUM? 100000
NUMBER OF YEARS? 7

RATE 10%
EACH YEAR      7YEARS

 50000         521794
 60000         626153
 70000         730512
 80000         834871
 90000         939229
 100000        1043588

INTEREST RATE %?
```

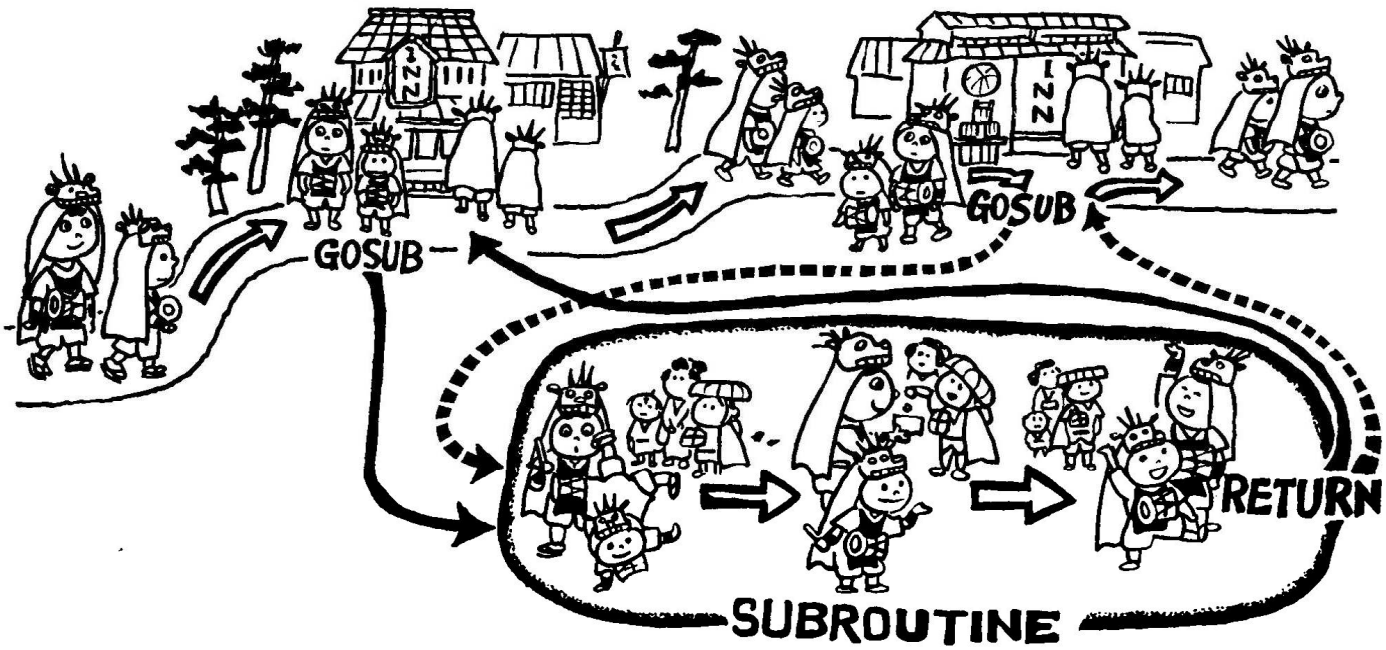
Subroutine is the Ace of Programs

In any program, jobs in the same procedure are repeated. Such jobs are summarized as a sub-program which can jump anytime, anywhere whenever required by the main program. This sub-program is called a "**subroutine**", for which the GOSUB statement is used. The following is an example of a program using the GOSUB statement.




```
10 PRINT "■"
20 PRINT TAB (10) ;"＊＊ TOTAL SALES ＊＊" : PRINT
30 PRINT "          5      10      15      20      25      30      35"
40 PRINT "┌┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┐"
50 PRINT "├┤     FLOUR " : GOSUB 200
60 PRINT "├┤     SUGER " : GOSUB 200
70 PRINT "├┤     WINE "  : GOSUB 200
80 PRINT "├┤     SAUCE" : GOSUB 200
110 PRINT "└┘"
120 PRINT : END
200 PRINT "├┤" ; : READ A
210 FOR N = 1 TO A : PRINT "▩" ; : NEXT N
220 PRINT A
230 RETURN
240 DATA 20, 15, 21, 24
```

In the above program, subroutines are statement numbers 200 through 230. By the RETURN statement at the end of subroutines, the program execution returns to the main program.



Stop, Check and Continue

The computer does not always work as desired when operated with a program generated. This requires a STOP statement to be inserted to check the contents of variables at the stop position. For example, in the following program, the STOP statement is inserted.

```
10 READ A, B
20 X = A * B
30 STOP
40 Y = A/B
50 END
60 PRINT X, Y
70 DATA 15, 5
80 END
RUN
BREAK IN 30
```

At this time, the display of variables is made in direct mode as follows:

```
PRINT A, B, X  CR
```

This enables you to check the program. To re-start the program, give a command to the computer as follows:

```
CONT  CR
```

The CONT Command is used when ;

- Program execution is stopped with the SHIFT – BREAK keys.
- Program execution is stopped by the STOP or END statement.
- Inputs are stopped at the INPUT statement using the SHIFT – BREAK keys.

The CONT Command cannot be Used ;

- Before program has been executed using the RUN command.
- When program is edited after execution has been stopped.
- If an error occurs during execution. Program returns to the "READY".
- To stop cassette tape operation, cassette tape operation is stopped with the BREAK key.
- When the MUSIC command for music sound is stopped.

The computer restarts execution from the stop position. With the END statement at statement number 50, the computer displays the READY and stops again. Then, print in the direct mode, as follows:

```
X = 3 : Y = 5  CR
```

The computer will then continue program execution when the CONT command is given, displaying 3 and 5 for variables X and Y.
The following program continues to display a triangle of * marks, indefinitely.

```
10 A = 0 : B = 38 : C = 1
20 FOR X = A TO B STEP C
30 FOR Y = 0 TO X
40 PRINT " * ";
50 NEXT Y : PRINT : NEXT X
60 K = A : A = B : B = K
70 C = –C : GOTO 20
```

With the BREAK key pressed while holding down the SHIFT key, program execution stops. Then, insert the following in the program and give the CONT command.

```
100 END
```

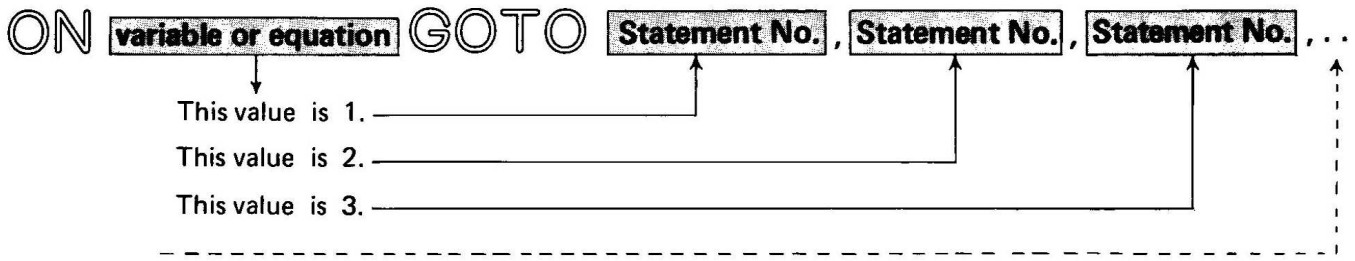
This is followed by the display below.

```
CONT ERROR
```

The CONT command cannot be used when a program is edited using a statement number after program execution has been stopped with the STOP statement, END statement or BREAK key operation. This requires special attention.

Jump en masse Using the ON ... GOTO Statement

You have learnt much about the GOTO statement. Description here is given of the ON . . . . . GOTO statement, an extended function of the GOTO statement.



For example, when the value of a variable or equation after ON is 3, a jump is effected to the third statement number that follows GOTO. In other words, it is possible to assign the branch of a program in accordance with the values of variables.

```
10 INPUT "NUMBER (1–3) ?" ; A
20 ON A GOTO 50, 60, 70
50 PRINT " X X X " : GOTO 10
60 PRINT " Y Y Y " : GOTO 10
70 PRINT " Z Z Z " : GOTO 10
RUN
NUMBER (1–3) ? 1
X X X
NUMBER (1–3) ? 2
Y Y Y
NUMBER (1–3) ? ■
```

Given 1.2, for example, integer 1 is processed, cutting off any place of decimals.

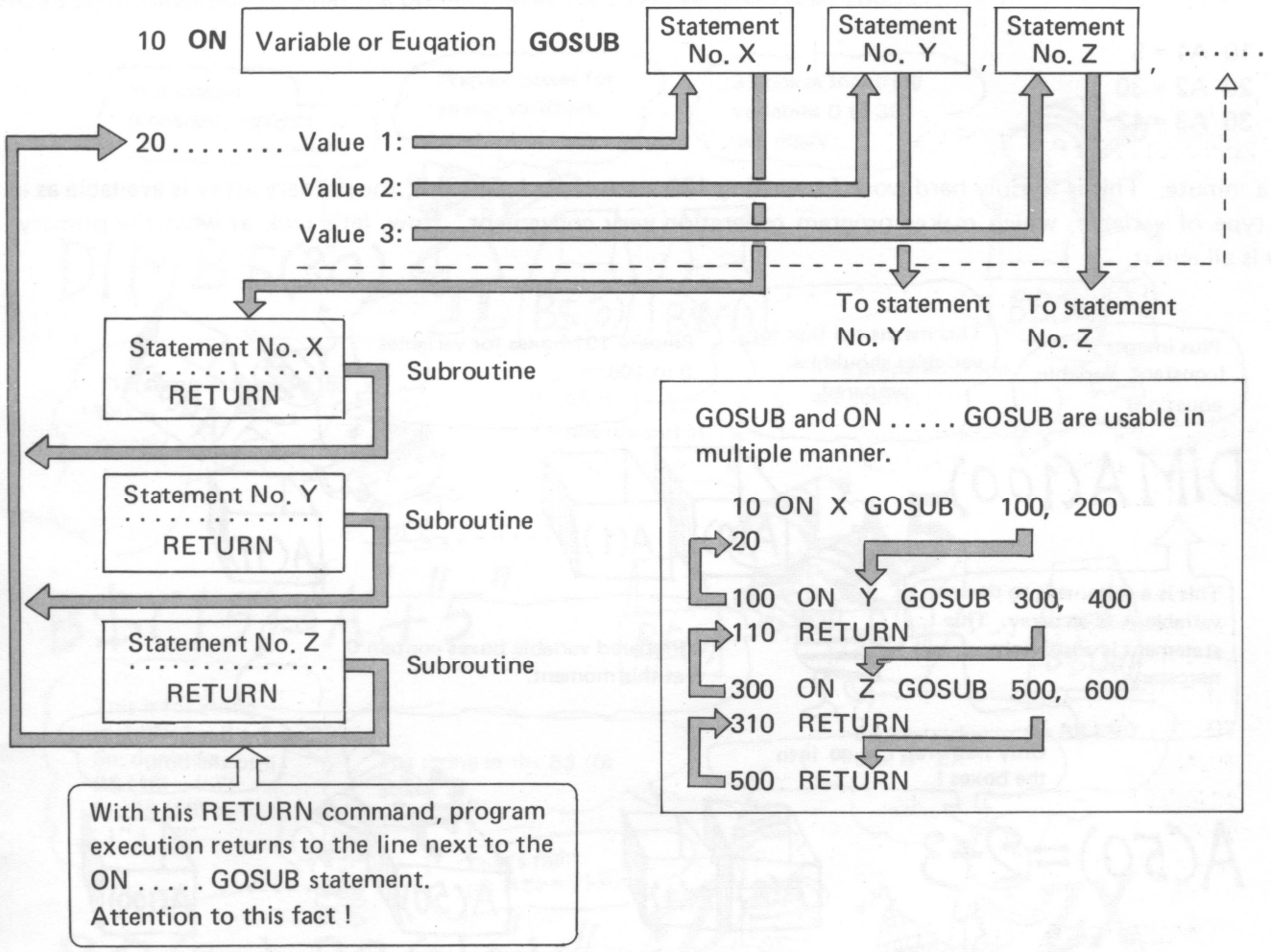
Let's play a joker-pick game using the ON . . . .GOTO statement. A joker is included in 5 cards. The place of the joker is unknown, of course. Guess where the joker is to compete with the computer for a win. When asked "Do you pass? " in the program, key-in 1 for pass, 2 for not pass and 3 if the game is no to be played. Three passes are allowed.

```
10 R = INT (5 * RND (1)) + 1
20 N = N + 1 : IF N = 6 THEN 120
30 INPUT "DO YOU PASS ? " ; X
40 ON X GOTO 60, 90, 50
50 PRINT "GAME NOT PLAYED !!!" : GOTO 120
60 NP = NP + 1
70 IF NP >= 4 THEN NP = NP–1 : N = N–1 : PRINT " NO PASS ALLOWED "
80 GOTO 20
90 NR = NR + 1
100 IF R = NR + NP THEN PRINT "LUCKY ! YOU HAVE SELECTED THE JOKER" : GOTO 120
110 PRINT "UNLUCKY ! YOU HAVE NOT SELECTED THE JOKER" : GOTO 20
120 END
```

Refer to page 70 for RND

ON ... GOSUB is the Use of a Subroutine Group

The ON . . . . . GOSUB statement is very similar in function to the ON . . . . . GOTO statement.



GOSUB and ON . . . . . GOSUB are usable in multiple manner.

```
10 ON X GOSUB   100, 200
20
100 ON Y GOSUB 300, 400
110 RETURN
300 ON Z GOSUB 500, 600
310 RETURN
500 RETURN
```

Now, let's consider the program for a time table to check your progress. Most important in the following program is that subroutines are called at statement number 180, despite the jump made at statement number 90 to statement numbers 170 through 190 of subroutines.
Thus, the GOSUB and ON . . . . . GOSUB statements can be used in a convenient, multiple manner.

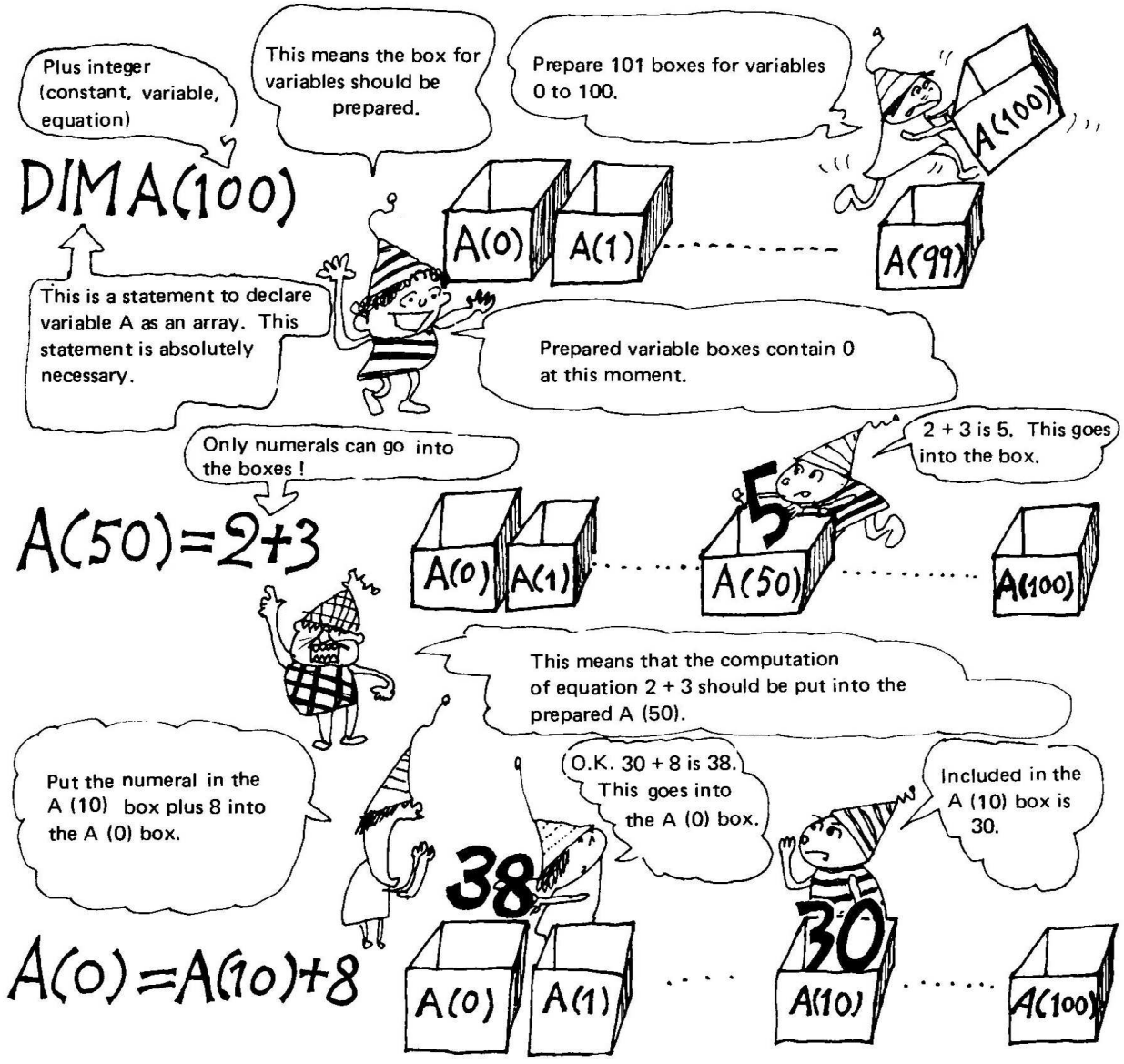
```
10 A$ = " FRENCH   " : B$ = "MATHEMATICS" : C$ = " ENGLISH "
20 D$ = "SCIENCE " : E$ = " MUSIC      " : F$ = "ATHLETICS"
30 G$ = "SOCIAL STUDIES" : H$ = "ART       " : I$ = "TECHNOLOGY "
40 J$ = "RELIGION   " : K$ = "ECONOMICS "
50 PRINT "WHAT DAY OF THE WEEK ?"
55 PRINT " (1 - MON, 2 - TUE, 3 - WED, 4 - THU, 5 - FRI, 0 - ALL)"
60 INPUT X$ : X = ASC (X$) - 47
70 FOR Y = 0 TO 3 : PRINT TAB (3 + 8*Y) ; Y + 1 ;
80 NEXT Y : PRINT
90 ON X GOSUB 170, 110, 120, 130, 140, 150
100 PRINT : GOTO 50
110 PRINT " MON : " ; A$ ; C$ ; D$ ; B$ : RETURN
120 PRINT "TUE : " ; H$ ; H$ ; E$ ; B$ : RETURN
130 PRINT " WED : " ; A$ ; C$ ; J$ ; K$ : RETURN
140 PRINT "THU : " ; D$ ; A$ ; E$ ; F$ : RETURN
150 PRINT "FRI : " ; A$ ; D$ ; I$ ; G$ : RETURN
170 FOR Y = 1 TO 5
180 ON Y GOSUB 110, 120, 130, 140, 150
190 PRINT : NEXT Y
200 RETURN
```

Primary Array has the Strength of 100 Men

Now, consider the substitution of variables for 100 items of data. The use of variables A1 and A2 makes the following possible.

```
10 A1 = 5
20 A2 = 30
30 A3 = 12
. . . . . .
```

Just a minute. This is terribly hard work for writing 100 statements ! For this, the primary array is available as a new type of variable, which makes program generation very convenient. Now, let's look at what the primary array is all about.



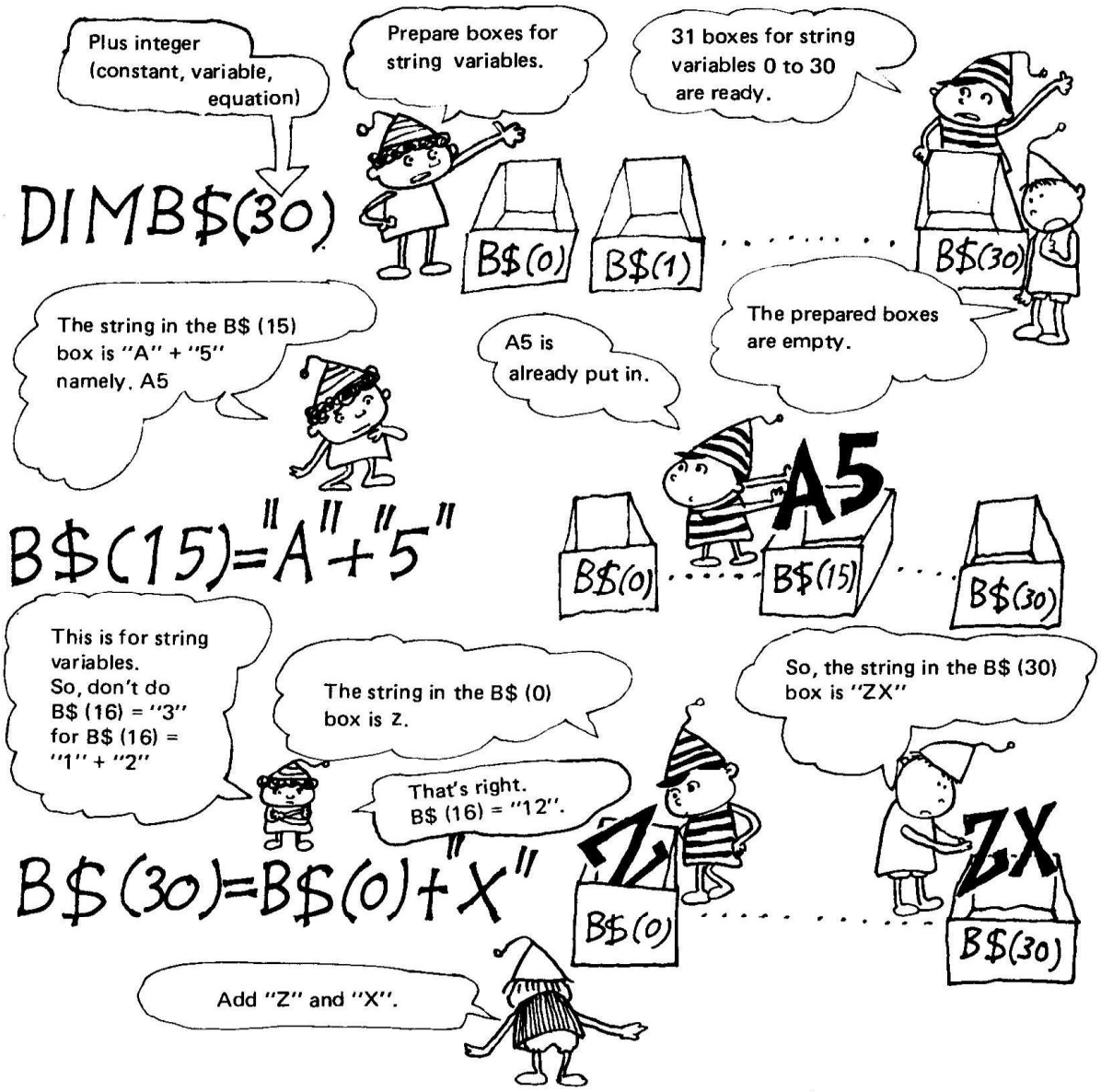
Now, you have understood what the primary array is, haven't you ?
Using the primary array, the program has been generated as follows:

```
10 DIM A (100)
20 FOR J = 1 TO 100
30 READ A (J)
40 NEXT J
50 DATA 5, 30, 12, . . . . . .
```

See, the program is very short. As is clear from this example, variables in the form of an array can assign the parenthesis of subscribed variables, such as A (J), with variable J. This is the main feature of the primary array.

Array is also Available for String Variables

Since an array is available for numeral variables, there must be an array available even for string variables. Here's an introduction to what the primary array for string variables is all about.



Let's generate a simple program. Just a look at this. Keeping variable strings in the form of arrays eliminates the labour of writing whenever they are used. The program itself is neat and simple.

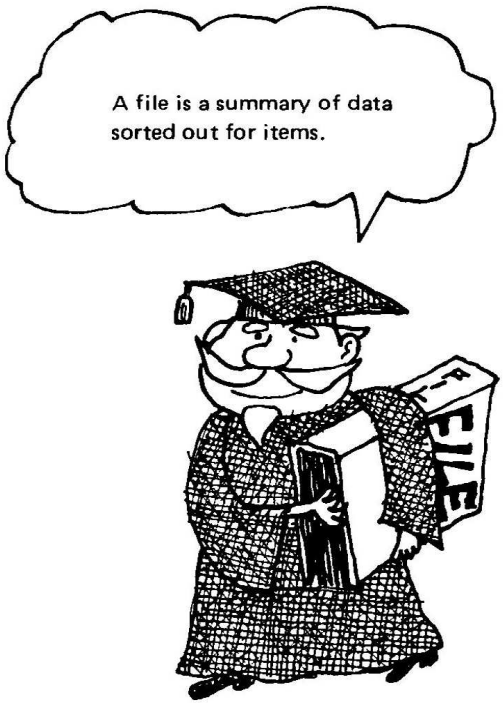
```
10 DIM A$ (2), B$ (2), C$ (2)
20 FOR J = 1 TO 2 : READ A$ (J), B$ (J)
30 C$ (J) = A$ (J) + "   " +B$ (J)
40 PRINT A$ (J), B$ (J), C$ (J)
50 NEXT J
60 END
70 DATA YOUNG, GIRL, WHITE, ROSE

RUN
YOUNG        GIRL        YOUNG GIRL
WHITE        ROSE        WHITE ROSE
READY
```

Array is the Master of File Generation (2)

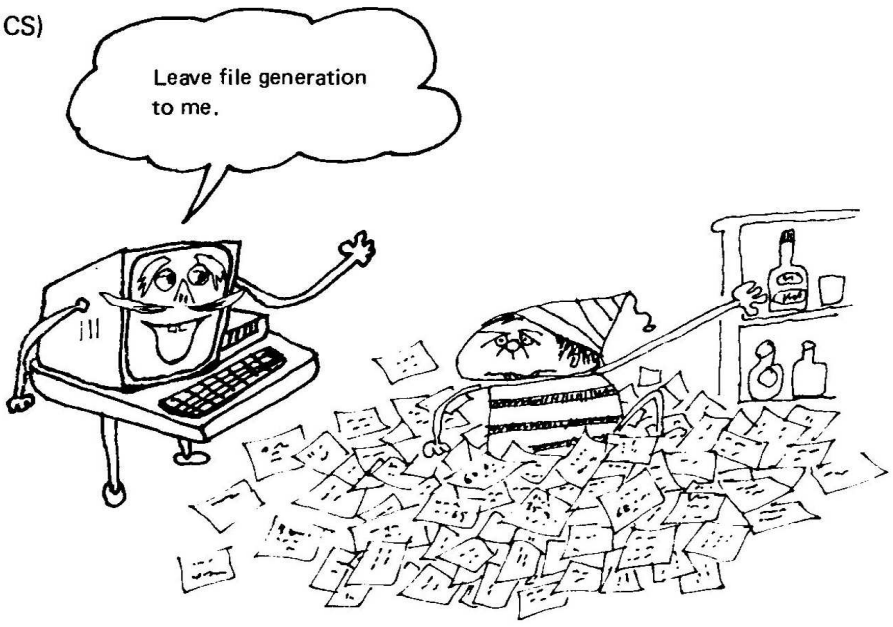
Some teachers say that testing is all right but putting test results in order is really hard. If so, some students insist testing should be stopped. A good method is available for teachers who are subject to giving tests to students. The use of an array helps them solve the problem ! The following shows student identification and marks for mathermatics.

Student No.	20	15	12	40	23	16	31	45	26	11
Marks	75	51	28	56	100	81	60	43	66	48

Generate a file program arranged in the order of merit.

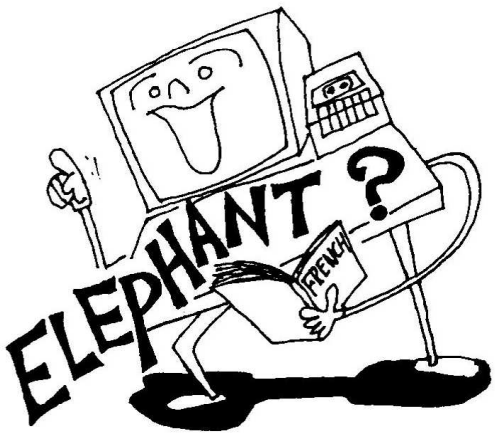



```
10 DIM A (10), B (10)
20 FOR J = 1 TO 10
30 READ A (J), B (J) : NEXT
40 FOR K = 1 TO 9 : M = 0
50 FOR J = K TO 10
60 IF B (J) < = M THEN 80
70 M = B (J) : L = J
80 NEXT J
90 B (L) = B (K) : B (K) = M
100 A1 = A (L) : A (L) = A (K) : A (K) = A1
120 NEXT K
130 PRINT "■"
140 PRINT "ORDER OF MERIT (MATHEMATICS)"
150 PRINT
160 PRINT "STUDENT NO. " ; TAB (14) ;
170 PRINT "MARKS"
180 FOR J = 1 TO 10
190 PRINT A (J) ; TAB (14) ; B (J) : NEXT J
200 END
210 DATA 20, 75, 15, 51, 12, 28, 40, 56, 23, 100
220 DATA 16, 81, 31, 60, 45, 43, 26, 66, 11, 48
RUN
ORDER OF MERIT (MATHEMATICS)

STUDENT NO.    MARKS
 23             100
 16             81
 20             75
 26             66
 31             60
 40             56
 15             51
 11             48
 45             43
 12             28
READY
```



Challenge of French Study

We used to study french words using word-notebooks. Smart and more simplified word-notebooks are available using the computer. French words and their meanings are contained in separate files. The computer gives two types of questions; one asking apout the meanings of Frech words retrieved from the file and the other asking English to be translated to French. In the program, the primary string array is used as the files containing French words and their meanings. Executing the following program, try to test your French vocabulary, answering a variety of questions the computer will ask you.




```
10 DIM A$ (10), B$ (10), C$ (10)
20 FOR J = 1 TO 10
30 READ A$ (J), B$ (J)
40 C$ (J) = A$ (J) +B$ (J)
50 NEXT J
60 K = INT (10 * RND (1) ) + 1
70 PRINT "■WHAT IS MEANING OF THE WORD ? "
80 PRINT A$ (K),
90 INPUT X$
100 AX$ = A$ (K) + X$
110 IF C$ (K) = AX$ THEN PRINT " O. K. !! " : FOR M = 1 TO 3000 : NEXT : GOTO 150
120 PRINT "WRONG " : FOR M = 1 TO 1000 : NEXT M
130 PRINT " ■ " ; SPC (10) : PRINT " ■■ " ; TAB (12) ; SPC (25)
140 PRINT " ■ " : GOTO 80
150 K = INT (10 * RND (1) ) + 1
160 PRINT "■TRANSLATE TO FRENCH"
170 PRINT B$ (K),
180 INPUT Y$
190 YB$ = Y$ + B$ (K)
200 IF C$ (K) = YB$ THEN PRINT" O. K. !! " : FOR M = 1 TO 3000 : NEXT M : GOTO 60
210 PRINT "WRONG " : FOR M = 1 TO 1000 : NEXT M
220 PRINT " ■ " ; SPC (10) : PRINT " ■■ " ; TAB (12) ; SPC (25)
230 PRINT " ■ " : GOTO 170
240 END
250 DATA CHAT, CAT, PORTE, DOOR, MAISON, HOUSE, CHIEN
260 DATA DOG, CANARD, DUCK, POISSON, FISH, MAIN, HAND
270 DATA FENETRE, WINDOW, FILLETTE, GIRL, FEMME
280 DATA WIFE
RUN
WHAT IS MEANING OF THE WORD ?
POISSON      ?
```

Refer to page 70 for RND

In this case, the question about the meaning of poisson is answeres by keying-in that English. Display of O.K. !! is on the TV screen to indicate you are correct. For any other answer, error display is made. Conversely, furthermore, there is the case when you answer "POISSON" when asked about translation.

Secondary Array is More Powerful

Let's look at this table (bottom right) which is an improvement on the test result table (bottom left) of mathematics, English and French for 3 students.

Name / Subject	John	Peter	Paul
Mathematics.	92	75	72
English	70	94	78
French	65	60	95

	Student	John	Peter	Paul
Subject	N / M	1	2	3
Mathematics	1	A (1, 1)	A (1, 2)	A (1, 3)
English	2	A (2, 1)	A (2, 2)	A (2, 3)
French	3	A (3, 1)	A (3, 2)	A (3, 3)

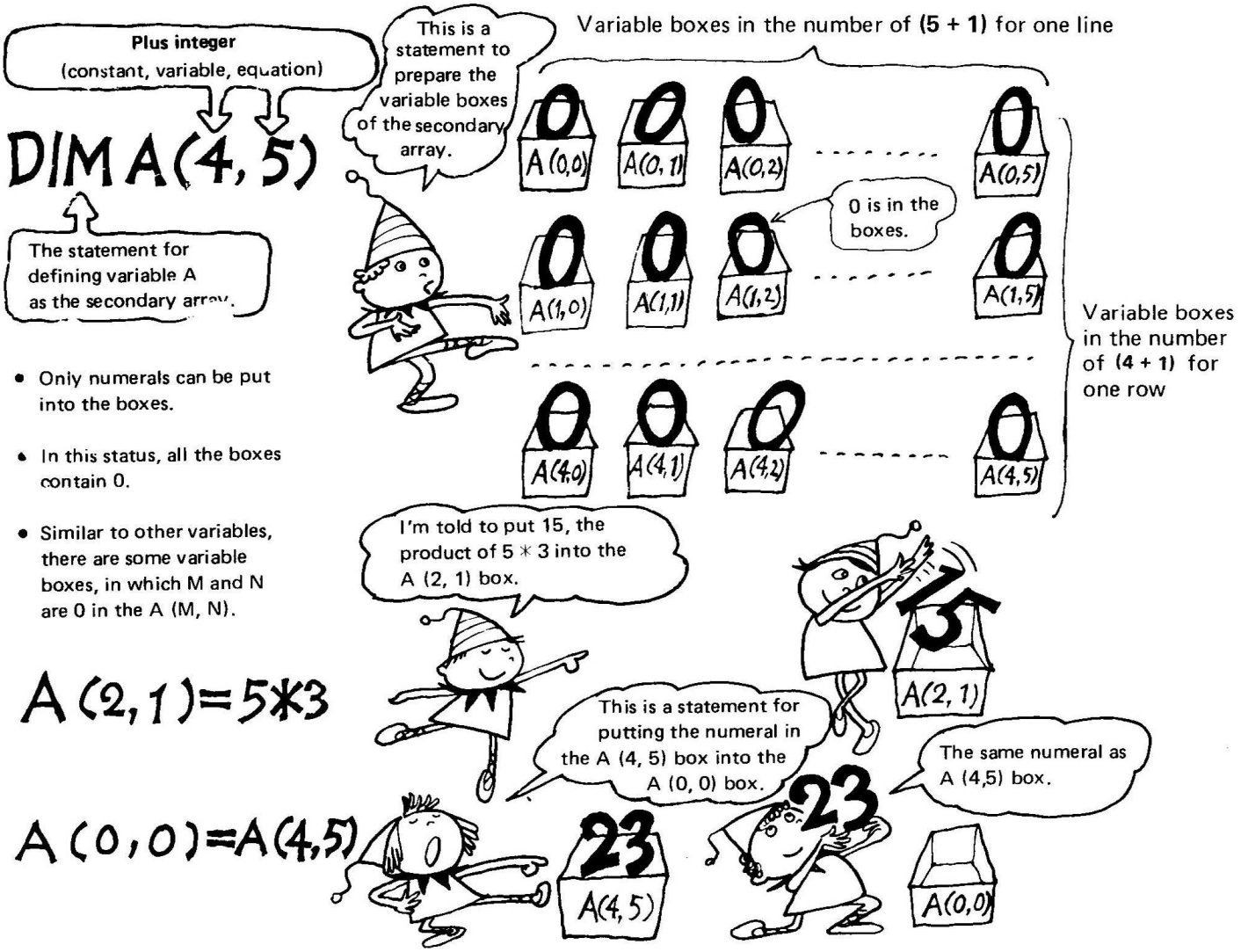
M = 1.... Mathematics
M = 2.... English
M = 3.... French

N = 1... John
N = 2... Peter
N = 3... Paul

In the table at right, the subject, student and marks are expressed as M (1 - 3), N (1 - 3) and A (M, N), respectively. This is very convenient, for example, as is evident in the following:

A (2, 3) ——→ M = 2 means English, N = 3 means Paul ——→ English marks of Paul

Simple ! Writing A (2, 3) alone gives a clear description of the English mark of Paul. M and N in the A (M, N) represent separate items. Writing A (M, N) using two items is called the secondary array. Two items used mean secondary array. The primary array previously described has one item.
Now, look at how this secondary array can be used in the program for the computer.



What About the Multiplication Table?

Have your children studied multiplication. The computer can help study. The multiplication table is to be generated, and can be used whenever required. The secondary array A (M, N) is used in the program. In other words, the value of M x N is assigned to A (M, N).

```
10 DIM A (9, 9)
20 PRINT "■"
30 FOR J = 1 TO 79
40 SET J, 6 : NEXT J
50 FOR J = 3 TO 27
60 SET 5, J : NEXT J
70 PRINT "■MULTIPLICATION"
80 PRINT
90 FOR J = 1 TO 9
100 PRINT TAB (4 * (J – 1) + 4) ; J ;
110 NEXT J : PRINT : PRINT
120 FOR M = 1 TO 9
130 PRINT M ;
140 FOR N = 1 TO 9
150 A (M, N) = M * N
160 PRINT TAB (4 * (N – 1) + 4) ;
170 PRINT A (M, N) ;
180 NEXT N
190 PRINT
200 NEXT M
210 END
RUN
MULTIPLICATION
```

Refer to page 76 for SET

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

READY

The secondary array uses many FOR . . . NEXT multiple loops as in this program. Therefore, it is suggested that you clearly understand the subscript in the secondary array. As is clear from multiplication, this is the 9 x 9 matrix. The secondary array plays an important role in processing data with matrixes and secondary elements.

Random Number is the One Left to Chance

Did you know that professional baseball pitchers use a random number table ? The random number sequence in the random number table is called a "progression" in which any numbers are equal in generation rates and the method of number generation has no rules (termed random). Such numbers used are called random numbers.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, ...

This progression has an equal rate for any number to be generated. In fact, however, the generation method is not random. In other words, they are not random numbers.

0, 1, 1, 1, 8, 7, 3, 9, 1, 6, 1, 2, 1, 1, 5, ...

This progression is rather random in generation. However, the generation rate of 1 is very high. Therefore, they are not random numbers, either.

Why not consider a random number sequence? You may have already noticed that this is not an easy job. Don't worry about it, for your computer does the job of generating random numbers. That's right, using the RND ( ) command. All you have to do is to put your favorite plus integer in the parenthesis. Any plus integer will be O.K.

```
10 FOR R = 1 TO 3
20 PRINT RND (2) ;
30 NEXT R
40 END
RUN
 0.38079483  0.75828109  0.44046507
READY
```

Because these are random numbers, the results are not always the same.

As is clear from the result, the values taken by the RND command are subject to the following:

$0 < \text{RND}(1) < 1$ **Attention to the fact that 0 and 1 are not included.**

Now, the mean for 1000 random numbers in taken, as follows:

```
10 FOR J = 1 TO 1000
20 R = RND (1) : S = S + R
30 NEXT J : RG = S/1000
40 PRINT RG
50 END
RUN
 0.4980582
READY
```

This mean is a value approximate to 0.5. From this fact, it is clear that the numbers thus generated are random.

Make a Dice Using the RND Function

Try to cast a dice. Naturally, one of the spots from 1 to 6 will turn up. The pleasure of playing a dice lies in the fact that any spots can turn up. In other words, spots from 1 to 6 are generated at random.
The computer is equipped with the RND function for random number generation. The problem is whether a dice can be produced by using this function. To tell you the truth, it can. How ?
First, multiply the RND function by six.

$$0 < \text{RND}(1) < 1 \xrightarrow{\times 6} 0 < 6 \times \text{RND}(1) < 6$$

Here you should remember the INT command. The INT command is used for the 6 x RND (1).

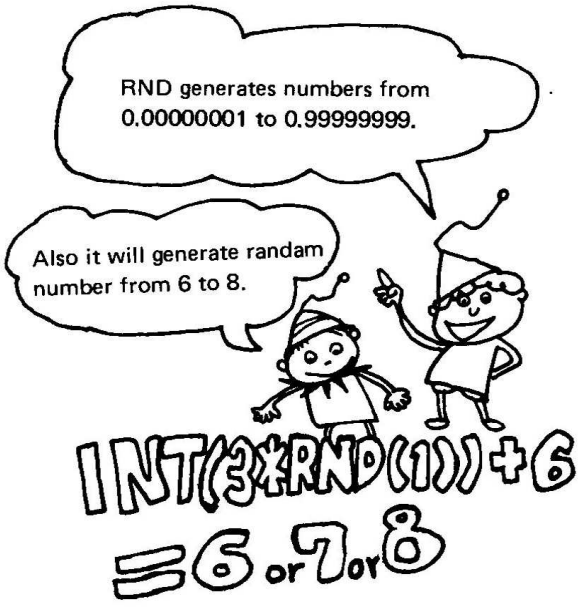
INT (6 * RND (1)) = 0, 1, 2, 3, 4, 5

With this, the integers from 0 to 5 are generated at random. The dice requires integers 1 to 6, and so they are not enough. Did you notice that 1 is further added to them ?

INT (6 * RND (1)) + 1 = 1, 2, 3, 4, 5, 6

Now, a dice is ready. Using this dice, let's check the rate, at which the spots turn up.

```
10 PRINT "NUMBER OF TIMES FOR DICE TO BE CAST"
20 INPUT N
30 FOR J = 1 TO N
40 R = INT (6*RND (1)) + 1
50 IF R = 1 THEN N1 = N1 + 1
60 IF R = 2 THEN N2 = N2 + 1
70 IF R = 3 THEN N3 = N3 + 1
80 IF R = 4 THEN N4 = N4 + 1
90 IF R = 5 THEN N5 = N5 + 1
100 IF R = 6 THEN N6 = N6 + 1
110 NEXT J
120 P1 = N1/N : P2 = N2/N : P3 = N3/N
130 P4 = N4/N : P5 = N5/N : P6 = N6/N
140 PRINT P1, P2, P3, P4, P5, P6
150 END
RUN
NUMBER OF TIMES FOR DICE TO BE CAST
? 5000
 0.1702      0.1654      0.1628      0.1626
 0.169       0.17
READY
```

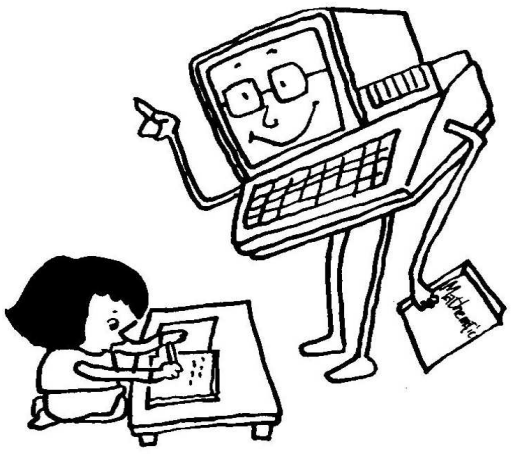


What do you think of the result ? The spots have almost the same rate of turning up. Mathematically, it is ideal that any spots turn up just once when the dice is cast 6 times. Therefore the ideal figures produced would be 1/6.

Quick Change into a Private Mathematics Teacher

Let's provide a mathematics teacher for your younger brother or sister. To begin with, exercise addition and multiplication of numerals from 1 to 9. For exercise of addition,

```
10 A = INT (RND (1) * 9 + 1)
20 B = INT (RND (1) * 9 + 1)
30 PRINT " " ; A : PRINT " + " ; B
40 PRINT " □□□□□□□ " : INPUT C
50 IF C = A + B THEN 80
60 PRINT "THINK HARD ONCE MORE"
70 GOTO 30
80 PRINT "OK! WELL DONE" : GOTO 10
```

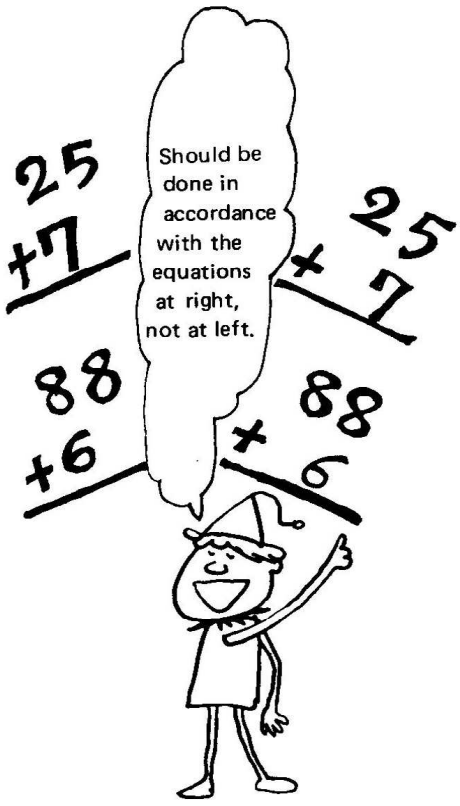


Using the random number at statement numbers 10 and 20, substitute two numerals from 1 to 9 for variables A and B. Then, the equation of adding two numerals is written, making an inquiry of an answer " ? ". At statement number 50, the judgement on whether the answer is correct or not is formed. For multiplication exercise, the " + " plus sign may be changed to the " * " sign at statement numbers 30 and 50.

Statement numbers 10 and 20 are changed as follows to use numberals from 1 to 99.

```
10 A = INT (RND (1) * 99 + 1)
20 B = INT (RND (1) * 99 + 1)
```

This seems to be correct. Actually, however, the number of 1 figure does not match the number of 2 figures (see diagram at right). So, something must be done to match the figures of the numbers. In other words, using the string processing facility, the figure count is checked to write an equation. Modify the previous program as follows to match the figures.




```
25 A $ = STR $ (A) : B $ = STR $ (B)
30 PRINT TAB (5 – LEN (A $)) ; A
35 PRINT " + " ; TAB (5 – LEN (B $)) ; B
40 PRINT " □□□□□□□ " : INPUT C
```

The computer converts two numerals to a string at statement number 25, and displays the respective numerals with their figures matched at statement numbers 30 and 40. Note how the TAB ( ) and LEN ( ) are used.

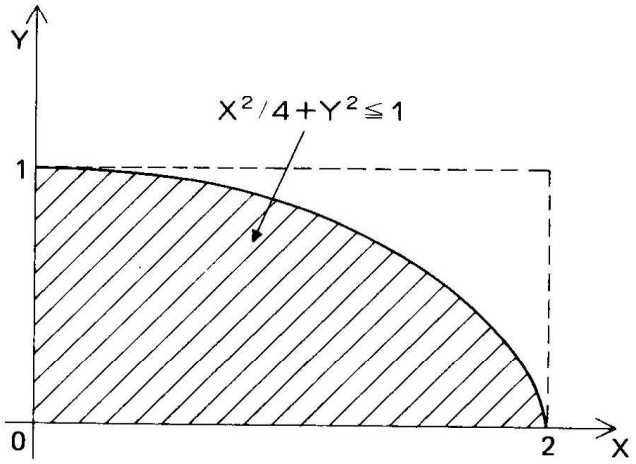
Tea Break

Now it is time to take a break. Key-in the program on the right and watch the random patterns being generated.

```
100 ? "■" ;
110 ? "■" : W = 1
120 FOR X = 1 TO 7 : FOR Y = 1 TO 5 : FOR Z = 1 TO 5
130 ? TAB ((X–1)    5) ; A$ ; : NEXT Z : ? : NEXT Y
140 A$ = CHR$ (INT (RND (1) * 223 + 33))
150 ? "■■■■■" ; : NEXT X : ? "■■■■" : W = W + 1
160 IF W > 4 THEN 110
170 GOTO 120
```

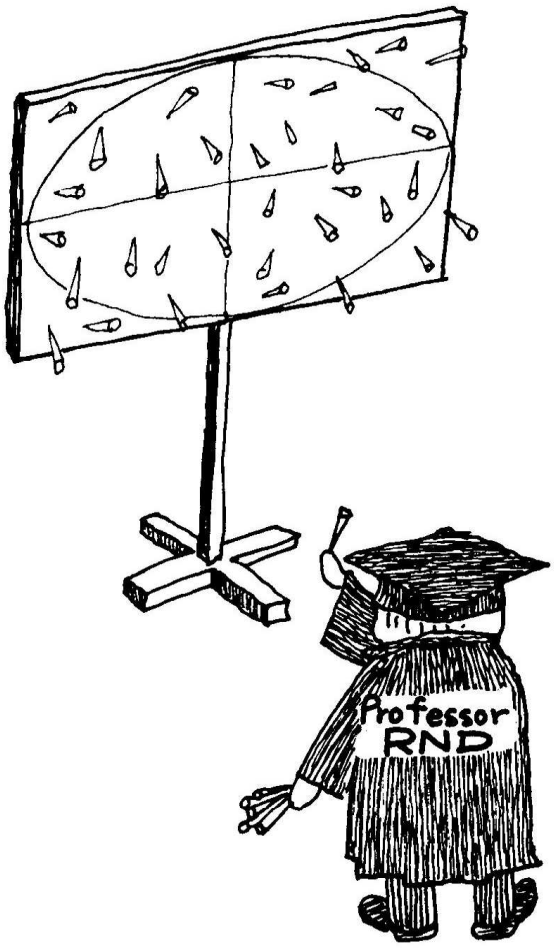
Probable Calculations for Figure's Areas

The use of random numbers makes integrals possible. In this case, probability is used, called the Monte-Carlo method numerical integral. Let's determine the area of an ellipse (definte intergral).
To simplify the calculation, 1/4 of an ellipse is considered. The hatching area in the diagram at right is an elliptical interior of $X^2/4 + Y^2 = 1$. Arrows are thrown at random to a rectangle made by the dotted lines. When many arrows are thrown, the ratio of hit times in the elliptical interior to the total thrown times is near that of the elliptical area to the rectangular area. The RND function takes a part of an arrow. The following is a program for the area of an ellipse.




```
10 PRINT "HOW MANY ARROWS TO BE THROWN ?"
20 INPUT N
30 PRINT "CALCULATION IN PROGRESS"
40 FOR J = 1 TO N
50 X = 2 * RND (1) : Y = RND (1)
60 IF X * X / 4 + Y * Y <= 1 THEN ND = ND + 1
70 NEXT J
80 S = 4 * (2 * ND/N)
90 PRINT "ELLIPTICAL AREA S = " ; S
100 END
```

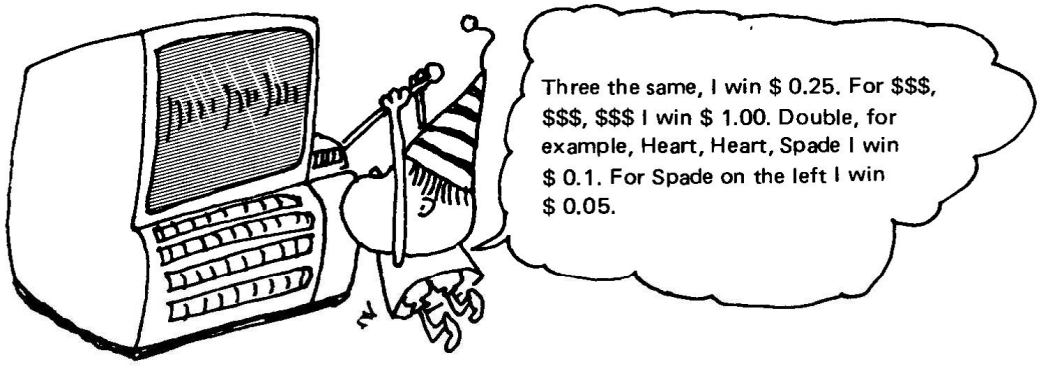
```
RUN
HOW MANY ARROWS TO BE THROWN ?
? 100
CALCULATION IN PROGRESS
ELLIPTICAL AREA S = 6.4
READY

RUN
HOW MANY ARROWS TO BE THROWN ?
? 1000
CALCULATION IN PROGRESS
ELLIPTICAL AREA S = 6.336
READY
```



The actual area of this ellipse is about 6.28. With 100 arrows thrown, the result is 6.4, which is very approximate to the actual area. With 1000 arrows fruther thrown, the result is more approximate to the actual area. Don' you find it interesting to determine such an area using random numbers ? Strange enough, the more random the numbers are, the more accurate the answer becomes.

Let's Make Money at the Casino ($$ Slot Machine $$)

Here in Las Vegas, the exciting city with dreams of entertainment and making a quick fortune. Michael, who is fond of gambling, is playing with a slot machine. With a 10 cent coin in, he pulls the lever down. Spades, Diamonds, Hearts, Clubs or Dollars appear in the three windows, and coins come out depending on the pattern combinations. The 10 cent coin is lost if he fails. Let's generate a program for that.

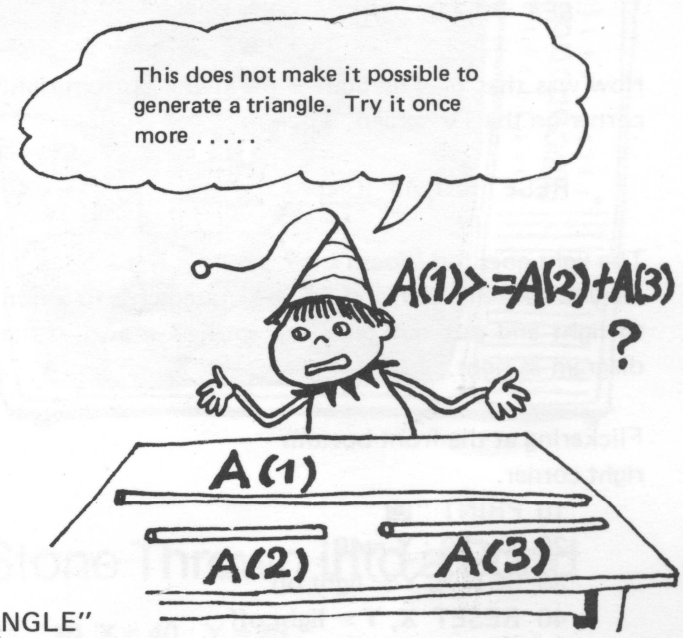



```
10 PRINT " ■ " : D = 0 : H$ = " ■■■■■■■■■ "
20 DIM N$ (5), A (3)
30 FOR X = 1 TO 10 : H$ = H$ + " ■■ " : NEXT X
40 FOR X = 1 TO 5 : READ N$ (X) : NEXT X
50 PRINT H$ ; " ■■■ $ = " ; SPC (5) ; " ■■■■■ " ; D
60 PRINT H$ ; " ■■ " : PRINT SPC (38) ; " ■ "
70 INPUT "FORCE TO PUSH THE LEVER ? " ; NN
80 FOR Y = 0 TO NN : RR = RND (1) : NEXT Y
90 FOR X = 1 TO 3 : A (X) = INT (5 * RND (1) ) + 1 : NEXT X
100 PRINT " ■■■■ " ;
110 FOR X = 1 TO 3 : PRINT N$ (A (X) ), : NEXT X : PRINT
120 IF A (1) <> A (2) THEN 170
130 IF A (2) <> A (3) THEN 160
140 IF A (1) = 5 THEN PRINT H$ ; "JACKPOT!! $1.00" ; SPC(15) : D = D + 1.00 GOTO 50
150 PRINT H$ ; "ALL THE SAME ! $0.25 " ; SPC (10) : D = D + 0.25 : GOTO 50
160 PRINT H$ ; "PRIZE IS $. 10 " ; SPC ( 5) : D = D + 0.1 : GOTO 50
170 IF A (1) = 1 THEN PRINT H$ ; "PRIZE IS $.05 " ; SPC (15) : D = D + 0.05 : GOTO 50
180 PRINT H$ ; "SORRY. 10 CENTS MORE PLEASE " : D = D - 0.1 : GOTO 50
190 DATA ♠♠♠ , ◆◆◆ , ♥♥♥ , ♣♣♣ , $ $ $
```

Did you win ? Like Michael, you are fond of gambling, so why not try to generate the shape of a solt machine, and insert it into the above program.

Let's Create Exercises using the RND Function

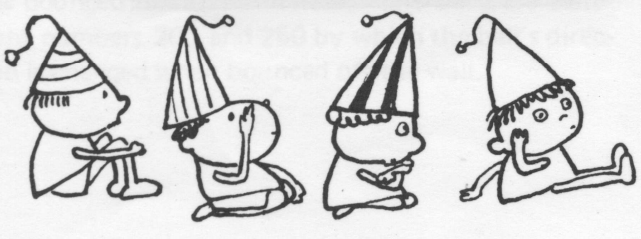
Let the RND function generate exercises for the determination of a triangular area. Here, the RND function your teacher. Since it provides the values of triangular sides using random numbers, you calculate the area. The RND function is the best partner for you.

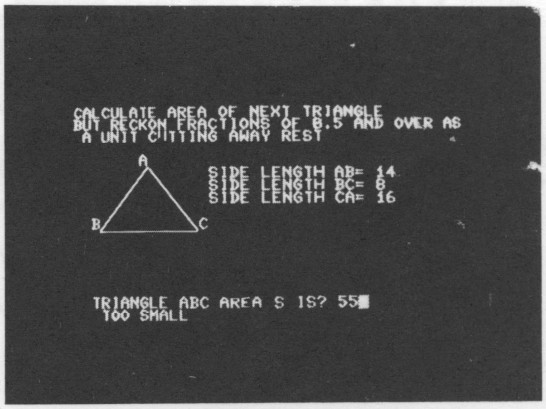



```
10 DIM A (3), L$ (4)
20 FOR J = 1 TO 4
30 READ L$ (J) : NEXT J
40 FOR J = 1 TO 3
50 A (J) = INT (20 * RND (1)) + 1
60 NEXT J
70 IF A (1) >= A (2) + A (3) GOTO 40
80 IF A (2) >= A (1) + A (3) GOTO 40
90 IF A (3) >= A (1) + A (2) GOTO 40
100 W = (A (1) + A (2) + A (3))/2
110 T = W : FOR J = 1 TO 3
120 T = T * (W – A (J)) : NEXT J
130 SS = SQR (T) : S = INT (SS)
140 IF SS – S >= 0.5 THEN S = S + 1
150 PRINT " C↓↓↓↓ "
160 PRINT " CALCULATE AREA OF NEXT TRIANGLE"
170 PRINT " BUT RECKON FRACTIONS OF 0.5 AND OVER AS A UNIT CUTTING AWAY REST"
180 PRINT
190 PRINT TAB (8) ; "A"
200 PRINT TAB (8) ; "◿◹"          ; TAB (15) ; L$ (1) ; A (1)
210 PRINT TAB (7) ; "◿    ◹"      ; TAB (15) ; L$ (2) ; A (2)
220 PRINT TAB (6) ; "◿        ◹"  ; TAB (15) ; L$ (3) ; A (3)
230 PRINT TAB (5) ; "◿            ◹"
240 PRINT TAB (3) ; "B ◿                ◹C"
250 PRINT TAB (4) ; "□□□□□□□□□□□ "
260 PRINT "↓↓↓"
270 PRINT TAB (3) ; L$ (4) ;
280 INPUT Y
290 IF Y = S THEN PRINT SPC (7) ; "O.K. !!" : FOR J = 1 TO 3000 : NEXT J : GOTO 40
300 IF Y < S THEN PRINT SPC (4) ; "TOO SMALL" : GOTO 320
310 PRINT SPC (4) ; "TOO LARGE"
320 PRINT " ↑↑ " ;
330 PRINT TAB (26) ; SPC (13) : PRINT " ↑ " ;
340 GOTO 270
350 DATA SIDE LENGTH AB =, SIDE LENGTH BC =
360 DATA SIDE LENGTH CA =, TRIANGLE ABC AREA S IS
```

Your Try Required !

If 0 or minus integer is placed in the parenthesis of the RND ( ), the random number values are obtainable only to a certain level. This is because the intitial values are set.





SET or RESET?

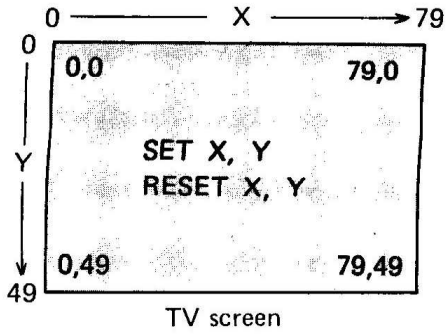
Using the direct mode, input the following statement:

SET 79, 49 CR

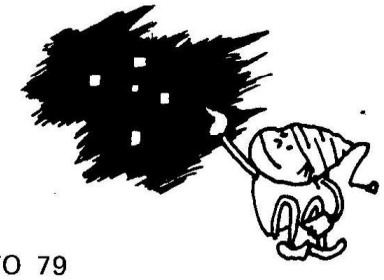
How was that ? The light is on at the bottom right corner on the TV screen. Then,

RESET 79, 49 CR

The light goes out, doesn't it ?
As you see, the SET and RESET statements function to light and put out only the spot as shown in the diagram at right.



TV screen



Flickering at the front bottom right corner.

```
10 PRINT "■"
20 X = 79 : Y = 49
30 SET X, Y ← light on
40 RESET X, Y ← light off
50 GOTO 30
```

TV screen is white.

```
10 PRINT "■"
20 FOR X = 0 TO 79
30 FOR Y = 0 TO 49
40 SET X, Y
50 NEXT Y, X
60 GOTO 10
```

For writing a rectangle of the screen size.

```
10 PRINT "■"
20 FOR X = 0 TO 79
30 SET X, 0
40 SET X, 49
50 NEXT X
60 FOR Y = 0 TO 49
70 SET 0, Y
80 SET 79, Y
90 NEXT Y
99 GOTO 99
```

For writing a slash line

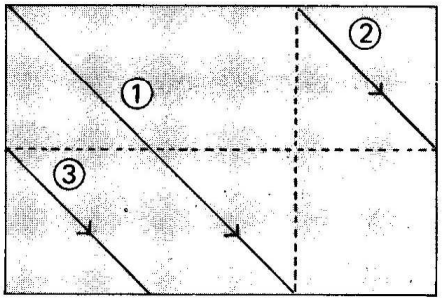
```
10 PRINT "■"
20 PRINT "LENGTH OF DIAGONAL";
30 INPUT Y
40 X = SQR (Y*Y/2)
50 FOR A = 1 TO X
60 SET A, 49 – A
70 NEXT A
80 GOTO 80
```

Pythagorean Theorem is applied.

Let's see something new !

```
10 PRINT "■"
20 FOR Z = 1 TO 99
30 SET Z, Z
40 NEXT Z
50 GOTO 50
```

TV screen after program execution



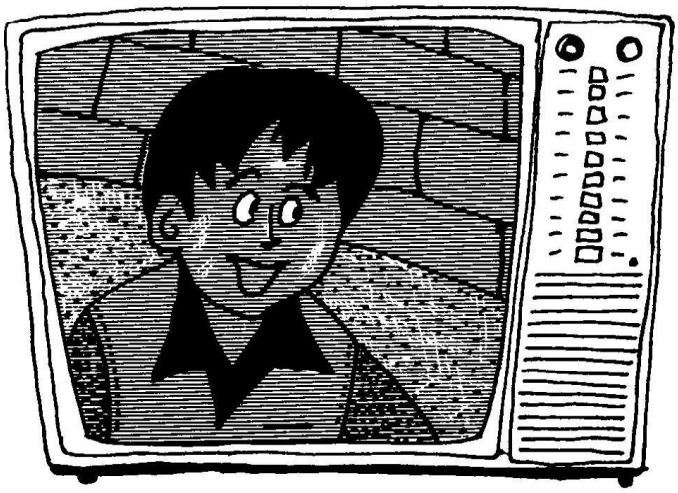
That's right. In fact, with more than 80 in X direction and more than 50 in Y direction,

```
X ← X – 80
Y ← Y – 50
```

This is automatically calculated.

Introduction to the Principles of TV

```
10 PRINT "■" : Y = 1
20 FOR X = 0 TO 79
30 SET X, Y
40 RESET X, Y – 1
50 NEXT X
60 Y = Y + 1
70 GOTO 20
```



Did you see the white line moving horizontally while shifting vertically one by one. This is designed to help you understand the principles of TV.
An ordinary TV has about 500 such lines to produce a picture, and about 30 pictures per second are transmitted.

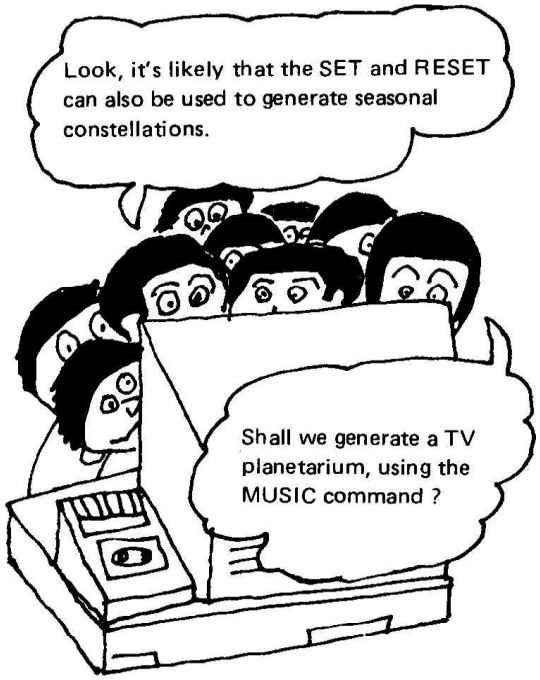
Ball Bounced .....

```
10 PRINT "■"
20 FOR X = 0 TO 79
30 SET X, 0 : SET X, 49
40 NEXT X
50 FOR Y = 0 TO 49
60 SET 0, Y : SET 79, Y
70 NEXT Y
80 X = 79 * RND (1) : Y = 49 * RND (1)
90 A = 1 : B = 1
100 SET X, Y
110 IF X < 2 GOSUB 200
120 IF X > 78 GOSUB 200
130 IF Y < 2 GOSUB 250
140 IF Y > 48 GOSUB 250
150 RESET X, Y
160 X = X + A : Y = Y + B : GOTO 100
200 A = – A : MUSIC " ¯A0 " : RETURN
250 B = – B : MUSIC " A0 " : RETURN
```

How's that ? Did you understand the program flow ? This program has three important points. The first is statement number 80 by which the starting point of a ball is generated using random numbers. The second are statement numbers 110 to 140 by which the check is made on whether the ball was bounced off the four walls. The third are statement numbers 200 and 250 by which the ball's direction is changed when bounced off the wall.

Stone Thrown into a Pond

```
10 X = 40 : Y = 25
20 DEF FNY (Z) = SQR (R*R – Z*Z)
30 PRINT "■" : SET X, Y
40 R = R + 5
50 FOR Z = 0 TO R
60 T = FNY (Z)
70 SET X + Z, Y + T
80 SET X + Z, Y – T
90 SET X – Z, Y + T
100 SET X – Z, Y – T
110 NEXT Z
120 IF R <> 25 THEN 40
130 GOTO 130
```



Wild Sketch (Rabbit and Fox)

Let's look at a model concerning the animal ecology on graphics display using the computer's SET statement. Taken up here is the model of animal struggle for existence. This model is ecologically given in the form of a differential equation. In fact, however, it is of prime use for reference as a simple example of numeric calcuation by the computer.

Rabbit and Fox

A rabbit and a fox are taken as an example of the relationship "The weak become the victim of the strong" in the animal kingdom. Also presumed here is that the fox lives on the grass-eating rabbit. According to the ecological model, the numbers of the respective rabbits and foxes, when presumed X and Y, change in accordance with the following differential equation:

$$\frac{dx}{dt} = AX - BXY \quad \text{. . . . . Change in the number of rabbits}$$

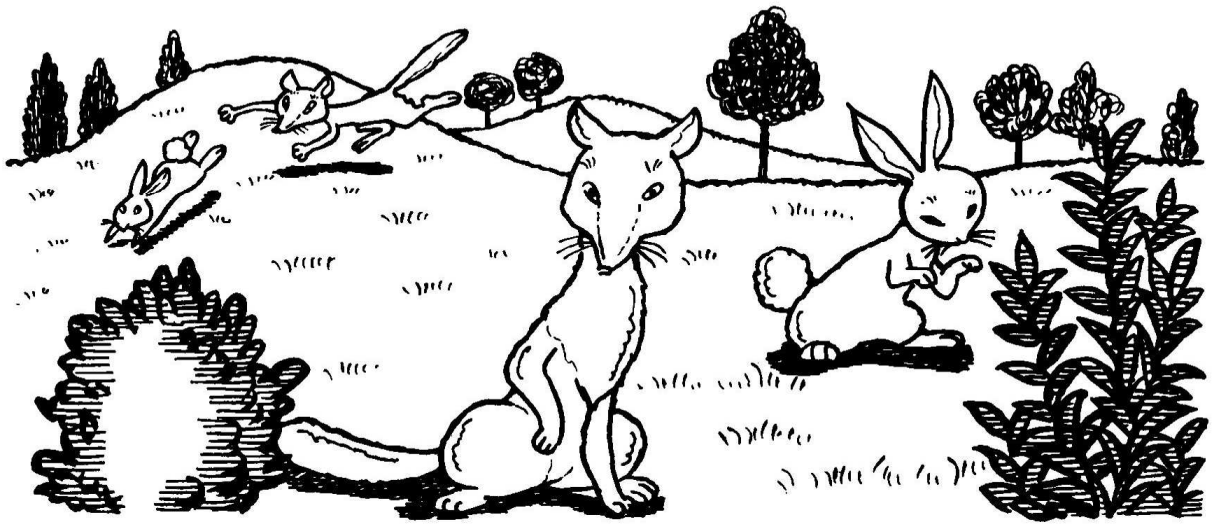
$$\frac{dy}{dt} = CXY - DY \quad \text{. . . . . Change in the number of foxes}$$

A, B, C and D are constants. The XY item represents the relationship of "The weak become the victim of the strong". Looking at the change in the number of rabbits, the second item would be zero if foxes were not in existence. The rabbits continue to increase by geometrical progression. In fact, however, foxes actually exist and increase in number to stop the increase of rabbits (2nd item). For the change in the number of foxes, the first item would be zero if rabbits were not in existence, and they are dying one after another. With rabbits, the number of foxes increases in accordance with the CXY value in the first item. This is how the rabbits and foxes are related to each other.

To look at this differential equation numerically using the computer, the approximate values based on the tangential method are utilized. This makes the derivative with respect to time (dt) approximate to the possible derivative time (DT), and when dx and dy are assumed to be DX and DY, this differential equation turns to an algebraical equation, as follows:

$$DX = (AX - BXY)\ DT$$
$$DY = (CXY - DY)\ DT$$

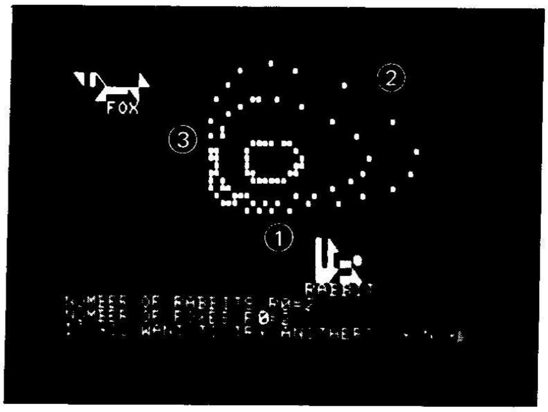
Shown on the following page is an example of the program based on this method.



The Secret of an Oval Graph (Rabbit and Fox)

```
10 REM RABBIT & FOX<===  REM executes nothing but comments on a program.
20 READ A, B, C, D, DT, TT, K
30 DATA 4, 2, 1, 3, 0.01, 2, 30
40 FOR H = 1 TO 20 : H$ = H$ + "↓" : NEXT H
50 PRINT "Ⓒ↓↓↓↓↓↓→◣▐◺ " ; SPC (3) ; " ◣ " : PRINT "→→→◢▀▀◣"
60 PRINT TAB (4) ; "FOX" ; "Ⓗ" ; H$ ; "↑↑↑" ; TAB (22) ; "▐◣"
70 PRINT TAB (22) ; "▐▐▄●" : PRINT TAB (22) ; "█◢▄◣"
80 PRINT TAB (21) ; "RABBIT"
100 PRINT " Ⓗ↓ " ; H$ ; SPC (38) : PRINT SPC (38) ; "Ⓗ" ; H$
110 INPUT "NUMBER OF RABBITS R0 = " ; X
120 INPUT "NUMBER OF FOXES F0 = " ; Y
200 FOR J = 1 TO K
210 FOR T = 0 TO TT/K STEP DT
220 X = X + (A * X - B * X * Y) * DT
230 Y = Y + (C * X * Y - D * Y) * DT
240 NEXT T
250 SET 4 * (X + 6), 4 * (8 - Y)
260 NEXT J
270 INPUT "DO YOU WANT TO TRY ANOTHER ? (Y/N) " ; Y$
280 PRINT "↑" ; SPC (38) ; "↑"
290 IF Y$ = "Y" THEN 100
300 END
```

Now for the program operation. The computer asks about the initial values for the numbers of rabbits and foxes. For example, key-in R0 =5 and F0 =5. Beginning with the initial values, both rabbits and foxes change in number with time, as seen on the TV screen.



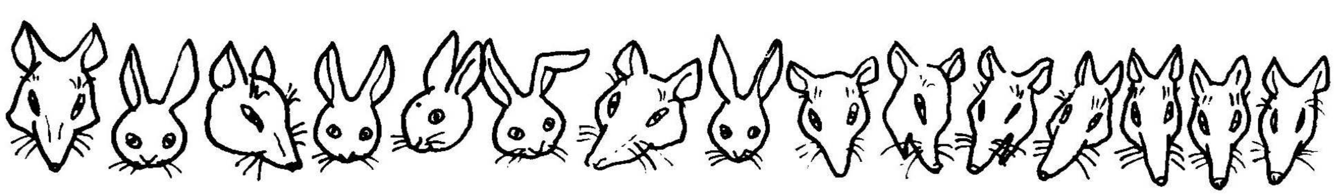
A fine oval figure is drawn. Let's consider the contents of the figure.

Part 1: With foxes being small in number at first, rabbits increase rapidly, while foxes increase gradually.

Part 2: With foxes increased in number, rabbits have reached the peak in number and began to decrease.

Part 3: Rabbits continue to decrease. So do foxes. Both rapidly decrease.

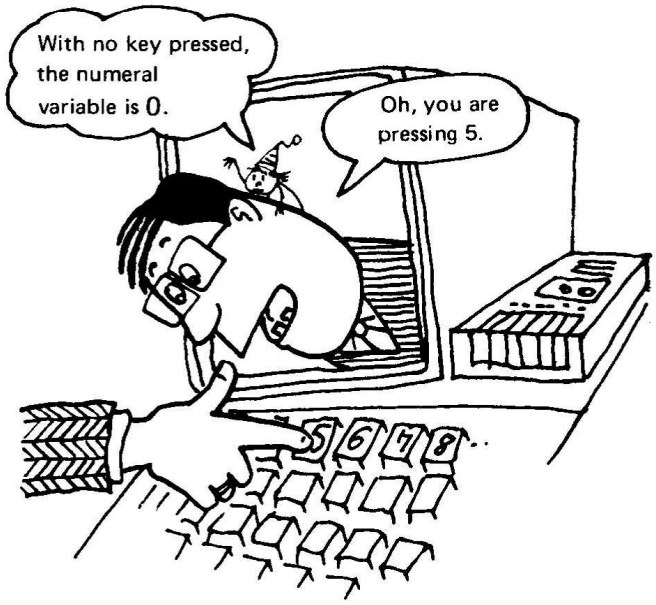
Thus, the number of the respective animals returns to the initial condition. Setting various initial values for the numbers of rabbits and foxes makes it possible to obtain a variety of oval graphs different in size.



GET is a useful Key Input

An introduction to a new type of statement for the data input from the keyboard is given here. Its Name

GET



This is most suitable for use in the execution of quick processing required for a game, for example. In the INPUT statement, "? " is asked, and with data keyed-in, the [CR] key is pressed for shift to the next step. Contrary to this, the GET command proceeds as follows:

- When a key is pressed while a program execution is under the control of the GET command, data for one character space is substituted for the variable for shift to the next step.
- When no key is pressed, 0 or blank is substituted for numeral variable or string variable for shift to the next step.

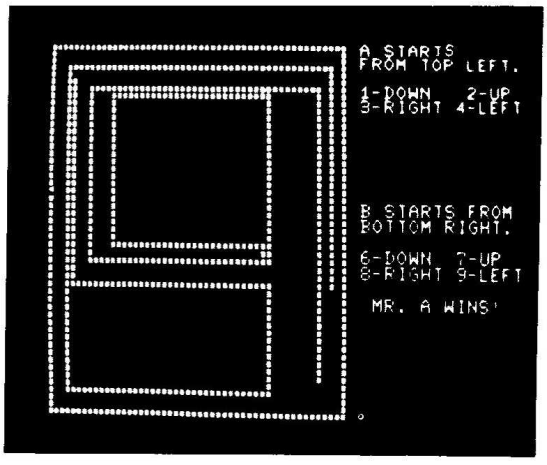
The GET command is often used in such a manner that the program waits until a character or numeral is keyed-in

Before playing a game using the GET command, let's work a simple exercise. Using the GOTO command, execute the GET command repeatedly. Try to key-in the numerals from 1 to 9 for variable Z.

```
10 GET Z
20 PRINT Z;:GOTO 10
```

When the key is pressed, the numeral of the key is displayed.

Now, let's generate a program for a position-taking game using the GET and SET commands. (See the program on the following page.) This game is for two players. Inside the square frame on the TV screen, two players start from the top left and bottom right positions, respectively, and block the movement of their opponent while changing directions with key operation as shown below. The player blocks the movement of its opponent to win the game.



Mr. A (Start from top left)

Keys [1] [2] [3] [4]

↓ ↑ → ←

Mr. B (Start from bottom right)

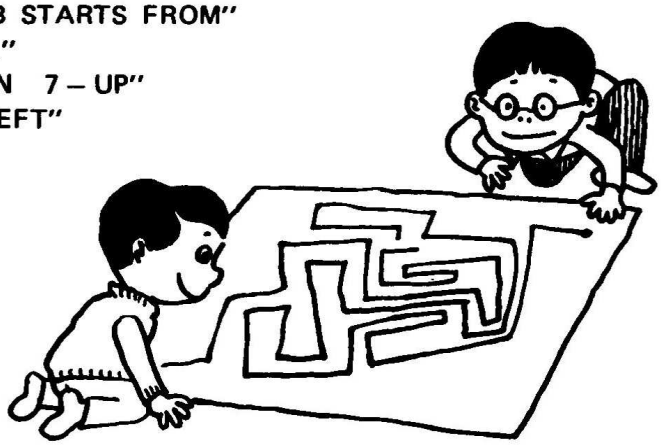
Keys [6] [7] [8] [9]

↓ ↑ → ←

While any key is pressed by the opponent, no other keys can be operated. Therefore, keys should be pressed alternately.

Let's Have a Look at a Position-Taking Game

This program uses a number of statements and commands you have so far learned. So why don't you look at how they are generated? In fact, however, this program requires a program area of almost 13K bytes.



```
10 CLR
20 DIM Z (49, 49)
30 PRINT " ■ " ; TAB (26) ; "A STARTS"
32 PRINT TAB (26) ; "FROM TOP LEFT."
35 PRINT " ↑ " : PRINT TAB (26) ; " 1 – DOWN  2 – UP"
40 PRINT TAB (26) ; "3 – RIGHT 4 – LEFT"
41 PRINT : PRINT
42 PRINT : PRINT : PRINT TAB (26) ; "B STARTS FROM"
45 PRINT TAB (26) ; "BOTTOM RIGHT."
50 PRINT : PRINT TAB (26) ; "6 – DOWN  7 – UP"
55 PRINT TAB (26) ; "8 – RIGHT 9 – LEFT"
60 FOR A = 0 TO 49
70 SET A, 0 : Z (A, 0) = –1
80 SET A, 49 : Z (A, 49) = –1
90 NEXT A
100 FOR A = 1 TO 49
110 SET 0, A : Z (0, A) = –1
120 SET 49, A : Z (49, A) = –1
130 NEXT A
140 X1 = 4 : Y1 = 4 : D1 = 1
150 X2 = 45 : Y2 = 45 : D2 = 7
160 GOSUB 200 : IF M = 1 GOTO 10
170 GOSUB 300 : IF M = 1 GOTO 10
180 GOTO 160
200 GET X
210 IF X = 0 THEN 260
220 IF X = 1 THEN Y1 = Y1 + 1 : GOTO 270
230 IF X = 2 THEN Y1 = Y1 – 1 : GOTO 270
240 IF X = 3 THEN X1 = X1 + 1 : GOTO 270
250 IF X = 4 THEN X1 = X1 – 1 : GOTO 270
260 X = D 1 : GOTO 220
270 D1 = X : IF Z (X1, Y1) = –1 THEN 400
280 MUSIC " □A0" : SET X1, Y1 : Z (X1, Y1) = –1
290 RETURN
300 GET X
310 IF X = 0 THEN 360
320 IF X = 6 THEN Y2 = Y2 + 1 : GOTO 370
330 IF X = 7 THEN Y2 = Y2 –1 : GOTO 370
340 IF X = 8 THEN X2 = X2 + 1 : GOTO 370
350 IF X = 9 THEN X2 = X2 – 1 : GOTO 370
360 X = D2 : GOTO 320
370 D2 = X : IF Z (X2, Y2) = –1 THEN 450
380 MUSIC "A0" : SET X2, Y2 : Z (X2, Y2) = –1
390 RETURN
400 PRINT TAB (27) ; "MR. B WINS ! "
410 MUSIC "C3DEG□CEG□CCDEG□CEG□CC " : M = 1 : RETURN
450 PRINT TAB (27) ; "MR. A WINS ! "
460 MUSIC "□G3□A□BCDE #FGAB□C□D□#D□E□F□#F□G " : M = 1
470 RETURN
```

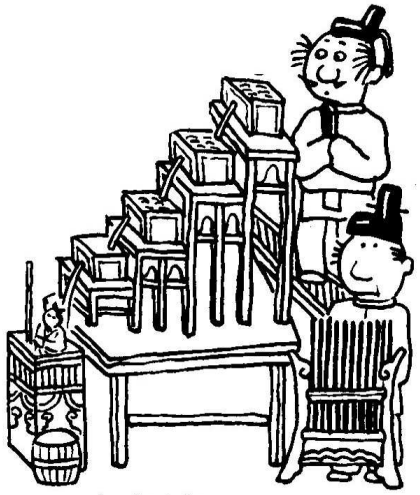
TI$ is a Digital Clock

It is interesting to have a clock function in the program. A digital clock or world clock depends on the program! Such a function can be fulfilled by TI$. Key-in the following on your computer and press the CR key.

```
PRINT TI$
```

The following will be on display, for example.

001234 ⇦ What is this?



Ancient Japanese Water Clock

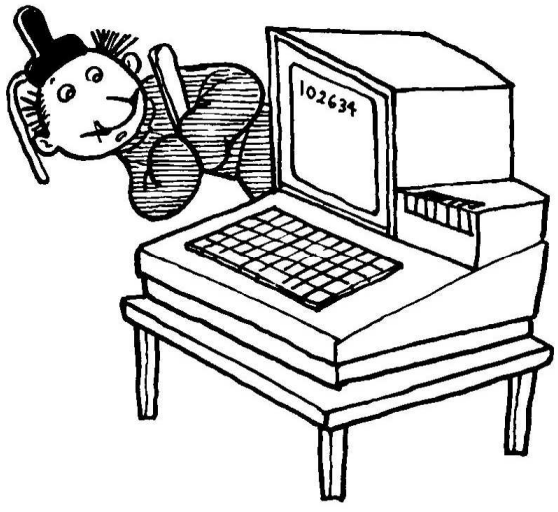
What does this number of 6 figures for the contents of string variable TI$ mean? That's right, it represents the time below:

00 12 34 ⇨ 00 hours, 12 minutes and 34 seconds

Hours Min. Sec.

This time does not always synchronize with your watch. For the TI$ is automatically set to 00 hours, 00 minutes and 00 seconds when the power switch of the computer is turned on. Display the TI$ once again, and you will notice the difference in contents from the previous display. That's correct, the number of 6 figures in the TI$ changes momentarily in the exactly same manner as the face of a digital clock. The value of the TI$ up to now is a lapse of time after the power switch was turned on.

```
10 TI$ = "102634"
20 PRINT TI$
30 TI$ = "256471"
40 PRINT TI$
50 END
RUN
102634
020511
READY
```



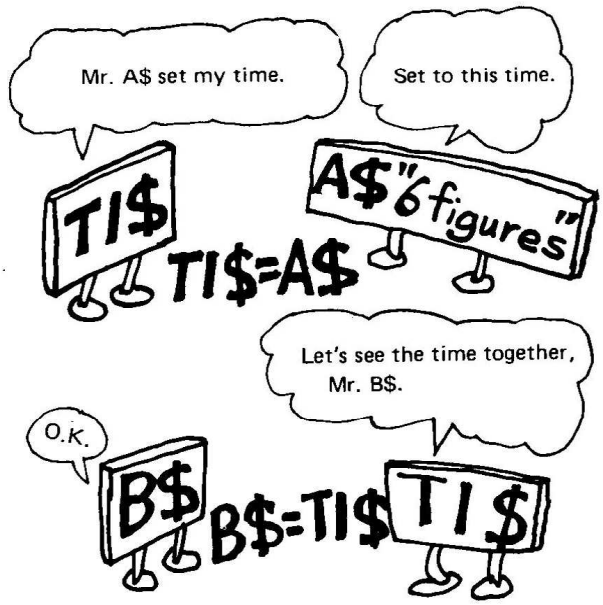
With the above statement numbers 10 and 30, the TI$ can be set to any particular time. However, the number between quotation marks " " should be only 6 figures. Statement number 30 indicates 25 hours, 64 minutes and 71 second, which should normally be 2 hours, 5 minutes and 11 seconds. Look at the PRINT result of the TI$ at statement number 30, showing 2 hours, 5 minutes and 11 seconds. This is how the computer automatically corrects the figures to make the correct time display.

Time for a Morning Call to a Friend in Tokyo?

The following program shows the relations between TI$ and ordinary strings, such as A$.

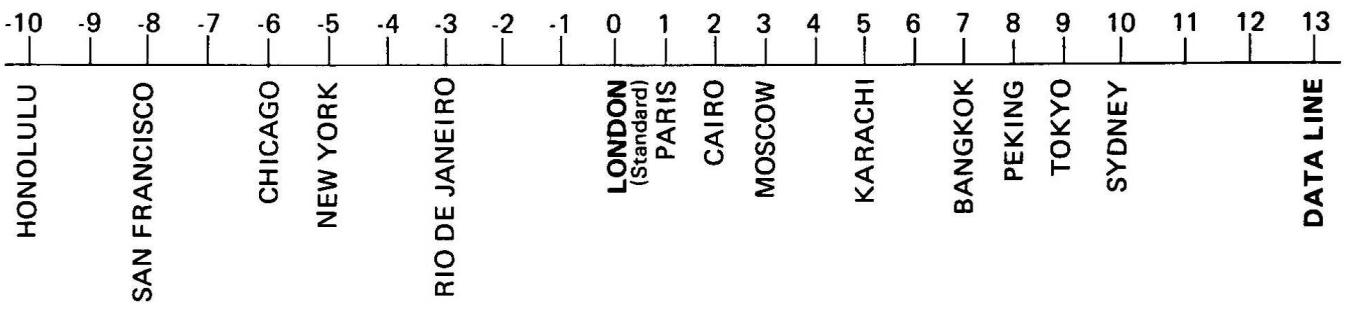
```
10 A$ ="123456"
20 TI$ = A$
30 PRINT TI$
40 FOR T = 1 TO 5000 : NEXT T
50 B$ = TI$
60 PRINT B$
70 END
RUN
123456
123500
READY
```

TI$ is a lapse of 4 seconds during program execution.



Here is a simple world clock. Of course, it's a digital clock. This can tell you the time in your favourite country. Now execute the following program, where the TI$ also ticks. There is a difference of 9 hours in time between London and Tokyo, isn't there? This is your turn to generate a program which tells the time of any other cities of the world.

```
10 PRINT "■"
20 DIM C$(10), D(10), E(10), T$(10)
30 FOR J = 1 TO 10 : READ C$ (J), D(J) : NEXT J
40 PRINT "WHAT TIME IS IT IN LONDON ? "
50 INPUT TI$ : PRINT "■"
60 PRINT "■" : T$ (1) = TI$
70 FOR J = 1 TO 10
80 E (J) = VAL (LEFT$ (T$ (1), 2)) + D (J)
85 IF E (J) = 24 THEN E (J) = 0
90 IF E (J) < 0 THEN E(J) = 24 + E(J)
100 T$(J) = STR$ (E(J)) + RIGHT$ (T$(1), 4)
110 IF LEN (T$(J)) = 5 THEN T$(J) = "0" + T$ (J)
120 PRINT C$(J) ; TAB(20) ; LEFT$ (T$(J), 2) ;
130 PRINT "H " ; MID$ (T$ (J), 3, 2) ; "M " ;
140 PRINT RIGHT$(T$ (J), 2) ; " S " : NEXT J
150 GOTO 60
160 DATA LONDON, 0, MOSCOW, 3, RIO DE JANEIRO, −3
170 DATA SYDNEY, 10, HONOLULU, −10, TOKYO, 9, CAIRO, 2
180 DATA NEW YORK, −5, SAN FRANCISCO, −8, PARIS, 1
```



DIFFERENCE IN TIME

Enjoyment of Music (A Visit to Mr. MZ-80K, a famous performer)

This is about the TEMPO and MUSIC statements that make the computer play music. The TEMPO statement assigns a tempo, as it is implied.
Musical notes are converted into strings according to the pitch (semitone and octave included) and duration of tones, and played under the MUSIC statement. Before describing in detail, the constitution of TEMPO and MUSIC statements are outlined below:

For tempo assignment	:	**TEMPO from the slow to fast tempo with numerals or variables from 1 to 7.**
For melody playing	:	**MUSIC with string variables to assign groups of notes.**
Note assignment	:	**Octave assignment** **# (Sharp)** **Scale** **Duration**

TEMPO X (X = 1 to 7): TEMPO Assignment

TEMPO X is a statement which assigns a tempo with X. When no TEMPO statement is assigned, the MUSIC statement is autmatically executed with TEMPO 4 assigned.

```
30  TEMPO 1    The slowest (Lento and Adagio)
30  TEMPO 4    Moderate (Moderate) : 4 times as fast as TEMPO 1
30  TEMPO 7    The fastest (Molto Allegro and Presto) : 7 times as fast as TEMPO 1.
```

MUSIC "String", M$ (String Variable)

This is a statement by which a melody or sound effect assigned by a string or string variable is actually produced from a speaker. Its tempo is in accordance with that assigned by the TEMPO statement.

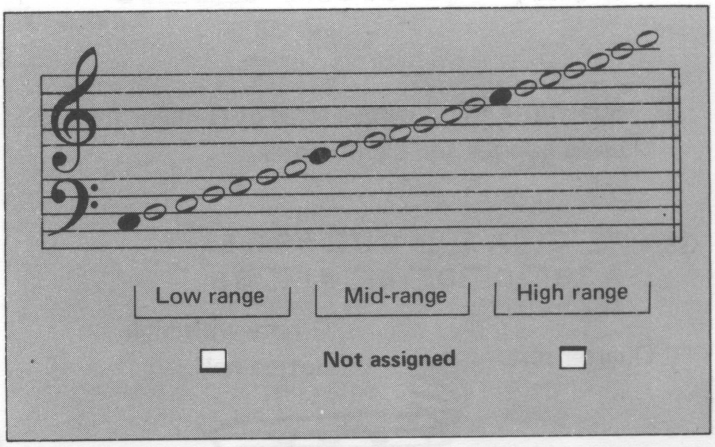
The following indicates how the melody or sound effect is converted into a string. Musical notes are assigned according to the tone pitch (octave and scale) and the duration (quarter note ♩ or eighth note ♪). Description is given of the assignments of the octave, scale and duration in that order.

Octave Assignment □ □

The sound range possible with the computer covers three octaves as shown at right. The black point shows a C note ("do" in C Major), and three C notes are separated by the octave assignment, as follows.

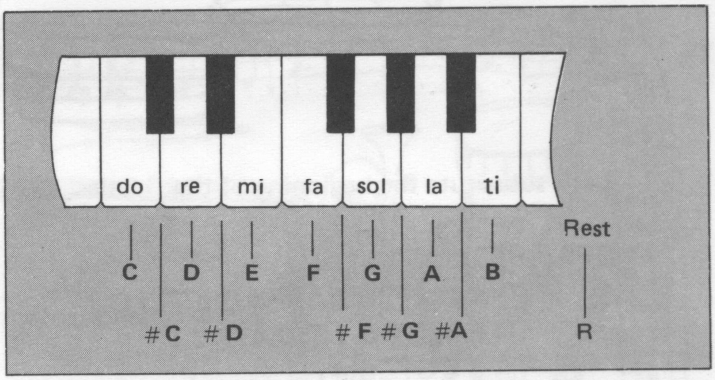
C in the low range ..... □C
C in the mid-range C
C in the high range ..... □C

□→  □→ 



Scale Assignment: CDEFGAB # F R

The CDEFGAB and # (sharp) symbol are used for scale assignment.
Relationship between the scale and CDEFGAB is shown in the diagram at right. The Sharp # symbol is used for semitone assignment.
The rest (no sound) is assigned with R.



Duration Assignment

Assignment is made on the duration of a tone whose pitch has already been assigned. Notes from thirty-second to whole are assigned with numerals from 0 to 9. (This assignment of duration also applies to R)

32nd rest	16th rest	Dot 16th rest	8th rest	Dot 8th rest	Quarter rest	Dot quarter rest	Half rest	Dot half rest	Whole rest
32nd note	16th note	Dot 16th note	8th note	Dot 8th note	Quarter note	Dot quarter note	Half note	Dot half note	Whole note
0	1	2	3	4	5	6	7	8	9

When notes identical in duration are repeated, the duration assignment from the second note can be omitted. With no duration assigned from the start, a program execution is carried out with quarter notes (duration 5) regarded to be assigned.

Prelude, Allegro and Amabile

Let's do some exercises of translating musical notes to strings. The following notes are translated referring to the table on the previous page.

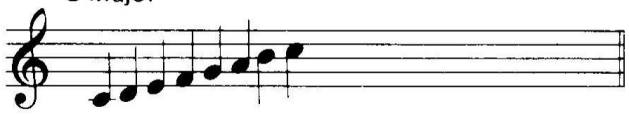
♪ First, the scales of C Major, D Major and E Flat (Quarter notes)

C Major "CDEFGAB□C"

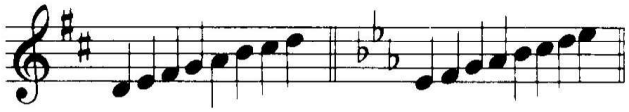
D Major "DE#FGAB□#C□D"

E Flat "#DFG#G#A□C□D□#D"

C Major

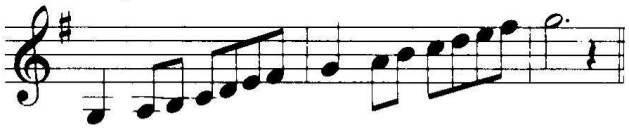


D Major E Flat



♪ Substitute the 2 octave scale of G major for G$ using quarter and eighth notes.

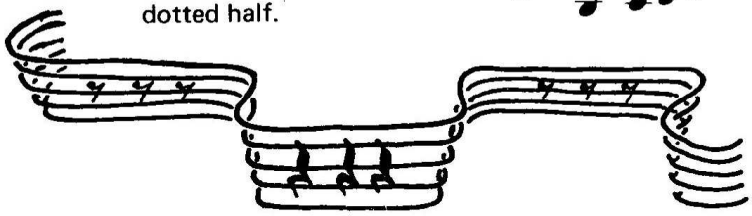
G Major



G$ = "□G□A 3□B C D E #F G 5
A 3B□C□D□E□#F□G 8
R 5"

Quarter rest.

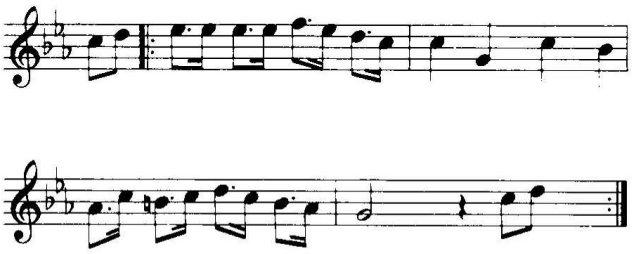
G note with high dotted half.



♪ Let's substitute the beginning of the Beatles' GIRL for GR$.

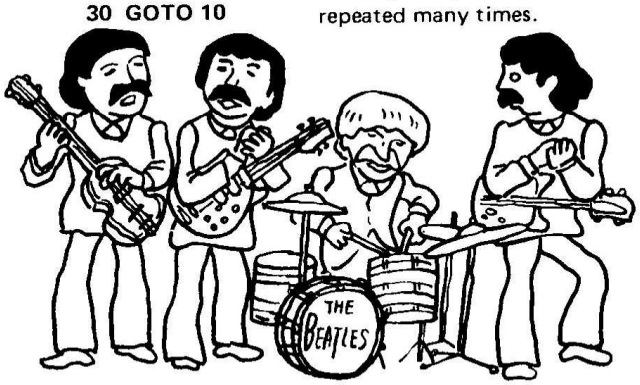
GR$ = "□C 3□D
□#D 4□#D 1□#D 4□#D 1
□F 4□#D 1□D 4□C 1
□C 5G□C#A
#G 4□C 1B 4□C 1
□D 4□C 1B 4#G 1
G 7R 5"

GIRL (John Lennon and Paul MaCartney)



```
10 TEMPO 5
20 MUSIC GR$
30 GOTO 10
```

The beginning of the "GIRL" is repeated many times.

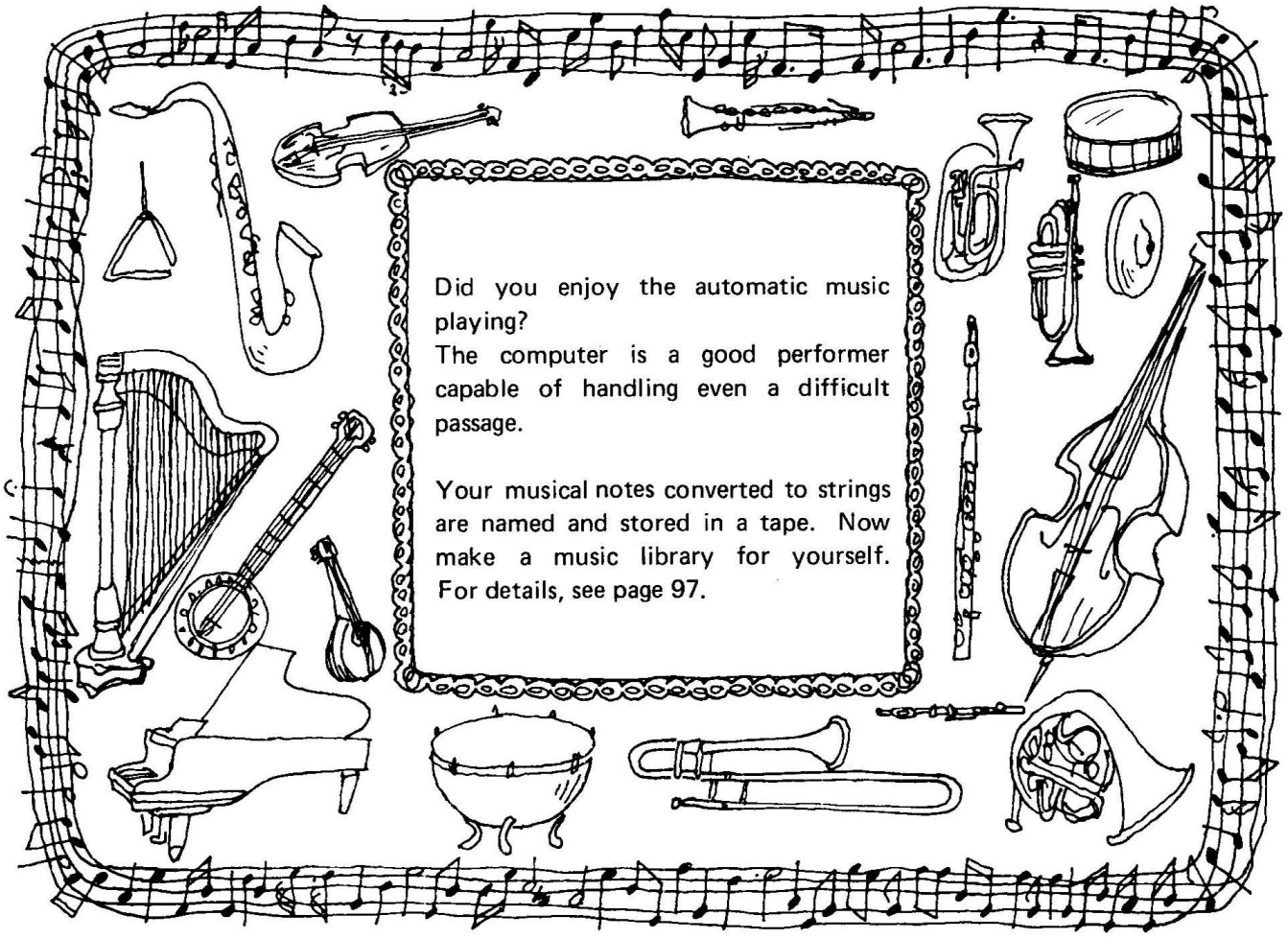


This seems to be difficult, but you will soon get used to it.
In case of a long tune, the string becomes too lengthy to be put into a statement of two lines on the TV screen. In this case, the tune is substituted by more than one string variables with their sum taken as a string variable for the MUSIC statement.

Now Make a Music Library

♪ Let's make a simple music box. This music box contains 2 small tunes, one is pleasant and the other, sad. Strings are properly cut for use in the box. Key in the J or N.

```
10 REM MUSIC BOX (FROM ETUDE OF F. KROEPSCH)
20 J1$ = " C 1 C E C G E C E C A D A F D A D "
30 J2$ = " G B B A G # F F D B C E C B G F D "
40 J3$ = " C B C G E C G E C 5 R "
50 N1$ = " D F A D F A G # A D G A # A A # C E G # A A G F F D A F "
60 N2$ = " D F A D F A E G # A E G # A A F D E # C A D # G A E F # C"
70 N3$ = " D D F A D E F # C D F A # C D 5 R 5 "
100 INPUT "MAJOR OR MINOR ? (J, N)" ; M$
110 IF M$ = " N " THEN 230
200 TEMPO 4
210 MUSIC J1$ ; J2$ ; J1$ ; J2$ ; J3$
220 GOTO 100
230 TEMPO 4
240 MUSIC N1$ ; N2$ ; N1$ ; N2$ ; N3$
250 GOTO 100
```



Did you enjoy the automatic music playing?
The computer is a good performer capable of handling even a difficult passage.

Your musical notes converted to strings are named and stored in a tape. Now make a music library for yourself. For details, see page 97.

I'll Get Up at 7 Tomorrow Morning

Alice is a heavy sleeper. Whenever she makes up her mind to get up early in the morning for jogging in the woods, she awakes to find it is time for school. She talked to her Prince about the fact, and then he advised her of a program saying, "Alice, why not set the timer so that you can get up at 7. Your favorite Chopin will sound then I will come down to the woods to wait for you."

```
10 REM JOHANN'S MAZURKA
20 MM$ = "A3" : M1$ = " A 5□#C 3□D□E□#F□G□#F 0□G□#F 4□E 3□D□#C B "
30 M2$ = " A 3□D 2 R 0□D 1□E 2□D□#C 3 B□#C 7□#C 3 "
40 M3$ = " A 3□#C 2 R 0□#C 1□D 2□#C B 3 A□D 7□D 3 "
100 PRINT " ■ " : INPUT " WHAT TIME IS IT NOW ? " ; TI$
110 INPUT "WHAT TIME DO YOU WISH TO WAKE UP ? " ; TM$
120 PRINT " ■ "
130 PRINT " ■↓↓↓↓↓ " ; TAB (17) ; TI$
140 IF VAL (LEFT$ (TI$, 6)) <> VAL (LEFT$ (TM$, 6)) THEN 130
150 TM$ = " 9999 "
160 TEMPO 3
170 MUSIC MM$, M1$, M2$, M1$, M3$, M1$, M2$, M1$, M3$
180 GOTO 120
```

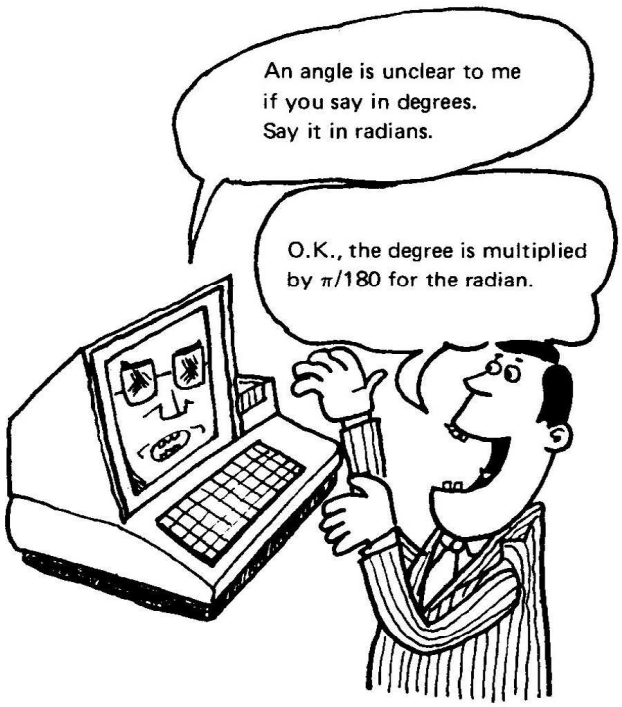
Input the actual time at the first INPUT statement. That's right, the actual time is substituted for TI$. As you remember, a number of 6 figures must be used, for example, 200000 for 8 o'clock at night. The CR key is pressed simultaneously with the time signal. Then, key-in the time you wish to wake up at. For example, 070000 is O.K. for Alice. You can create your own clock with a timer available to show the time anywhere in the world. This all depends on your ability of program generation.

Two Exercises

Determine the value and curve of SIN when an angle varies from 0 degree to 180 degrees with 10 degree increments.

```
10 PRINT "◙➡" ;TAB (5) ;"SIN"
20 PRINT
30 FOR K = 0 TO 180 STEP 10
40 X = K ✳ π/180
50 S = SIN (X)
60 A = INT (10 ✳ S)
70 PRINT K ; TAB (4) ;
80 PRINT S ; TAB (18 + A) ;
90 FOR J = 0 TO A
100 PRINT "✳" ;
110 NEXT J
120 PRINT
130 NEXT K
140 END
```

Value in degree unit is changed to radian.



Execute the above program. This graph is rather rough. As for drawing graphs, a lot of examples will be described later so that you can learn more about them.

Now, let's generate a program to determine the prime numbers. A prime number is the one that cannot be divided by any integer smaller than itself, except for 1. Since the first prime number is 2, the multiples of 2, namely, even numbers larger than 2 are not prime numbers. To use variables with subscripts effectively, even numbers are excluded from the start.

```
10 DIM P (255)
20 FOR J = 0 TO 255
30 P (J) = J ✳ 2 + 3 : NEXT J
40 FOR K = 0 TO SQR (255)
50 IF P (K) = 0 THEN 90
60 KK = K + P (K)
70 FOR L = KK TO 255 STEP P (K)
80 P (L) = 0 : NEXT L
90 NEXT K
100 PRINT 2 ;
110 FOR M = 0 TO 255
120 IF P (M) = 0 THEN 140
130 PRINT P (M) ;
140 NEXT M
150 END
```

Substitute 256 odd numbers from 3 to 513 for the parenthesis of variable P with a subscript.

Find prime numbers from the small in value and substutute 0 for the values of the multiples in the parenthesis of P.

The only even number "2" is first displayed, and then the values of P ( ) which are not 0, namely, prime numbers are on display.

This program is a bit complex, isn't it ? Note that the multiples of the prime number are excluded from the start.

Here's Advice on how Lists can be made

Names are sorted out when making a list of members. The use of a convenient program, if any, facilitates listing of any kind.

Here you learn how to sort strings for address books, telephone numbers or housekeeping account books.

```
10 PRINT " HOW MANY PERSONS ARE SORTED ? "
20 INPUT X
30 DIM N$ (X)
40 PRINT " KEY – IN NAMES ONE BY ONE "
50 PRINT " BUT IF 0, JOB DISCONTINUED ! "
60 FOR A = 1 TO X : A$ = STR$ (A)
70 PRINT " NAME PLEASE  " ; " ( " ; A$ ; " ) "
80 INPUT N$ (A)
90 IF N$ (A) = "0" THEN 110
100 NEXT A
110 A = A – 1
120 FOR B = 1 TO A – 1
130 FOR C = 1 TO A – B
140 D = LEN (N$ (C)) : E = LEN (N$ ( C + 1)) : F = 1 : IF D < E THEN E = D
142 X = ASC (MID$ (N$ (C), F, 1))
143 Y = ASC (MID$ (N$ (C + 1), F, 1)) : IF X > Y THEN 150
144 IF X < Y THEN 180
145 IF (E = F) * (D = E) THEN 180
146 IF (E = F) * (D > E) THEN 150
148 F = F + 1 : GOTO 142
150 K$ = N$ (C)
160 N$ (C) = N$ (C + 1)
170 N$ (C + 1) = K$
180 NEXT C, B
190 PRINT
200 FOR B = 1 TO A
210 PRINT N$ (B)
220 NEXT B
230 PRINT : END
```

Name is keyed-in.

The order is substituted.

Result is displayed.

Original List (Keyed-in)

TOM BROWN
HAROLD GREEN
JIM JONES
ANNE MILLER
TOM CARTER
ELICE THOMAS

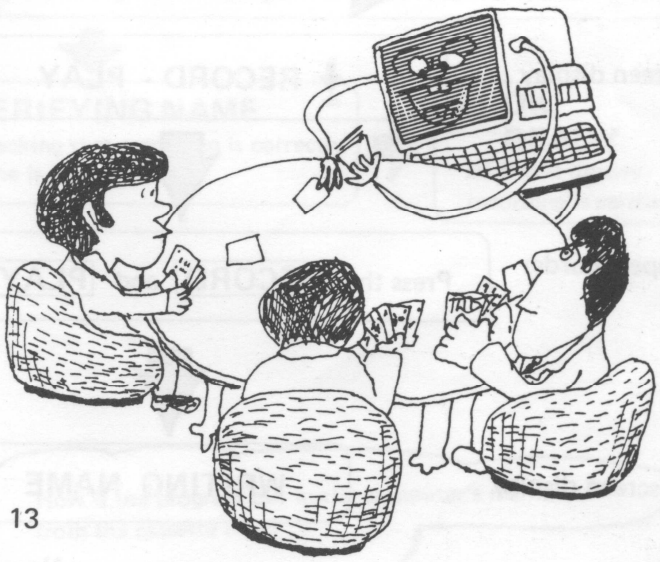
Sorted List

ANNE MILLER
ELICE THOMAS
HAROLD GREEN
JIM JONES
TOM BROWN
TOM CARTER

Cards if Dealt by a Poker Player

The computer deals cards for you. It shuffles them correctly using random numbers, causing no trickery to occur.

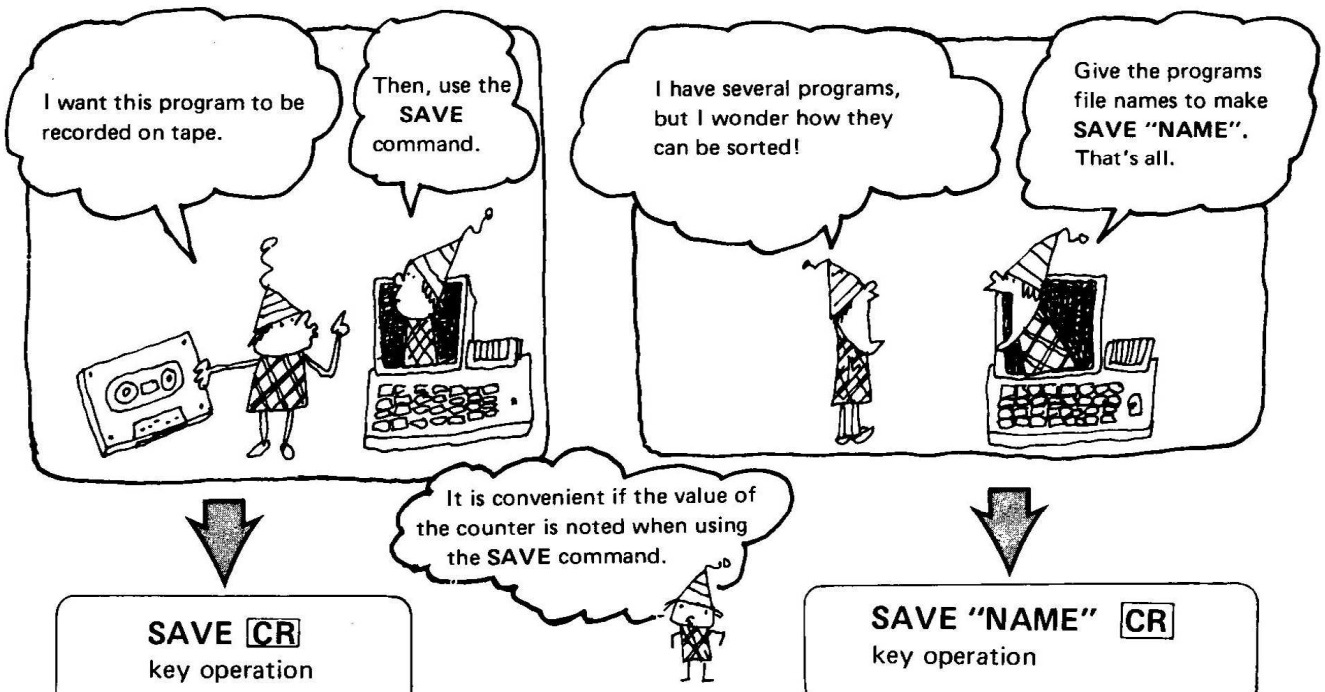
```
10 DIM X (4, 13)
20 C = 0
30 PRINT : FOR A = 1 TO 5
40 GOSUB 90 : PRINT : NEXT A : PRINT
50 PRINT " IS YOUR HAND ALRIGHT WITH THESE CARDS ? "
60 INPUT "ALL RIGHT (1), GIVE ME NEXT (2) ? " ; A
70 ON A GOTO 400, 30
80 GOTO 50
90 C = C + 1 : IF C = 51 THEN 500
100 M = INT (4 * RND (1)) +1
110 N = INT (13 * RND (1)) + 1
120 IF X (M, N) = – 1 THEN 100
130 X (M, N) = –1
140 IF N = 1 THEN PRINT " ACE  : " ; : GOTO 180
150 IF N = 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
160 IF N < 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
170 ON N – 10 GOTO 200, 210, 220
180 ON M GOTO 300, 310, 320, 330
200 PRINT " JACK  : " ; : GOTO 180
210 PRINT " QUEEN : " ; : GOTO 180
220 PRINT " KING  : " ; : GOTO 180
300 A$ = "♠" : GOTO 340
310 A$ = "♥" : GOTO 340
320 A$ = "◆" : GOTO 340
330 A$ = "♣" : GOTO 340
340 FOR B = 1 TO N
350 PRINT A$ ;
360 NEXT B
370 RETURN
400 PRINT
410 PRINT " THEN I RESHUFFLE. "
420 FOR M = 1 TO 4 : FOR N = 1 TO 13
430 X (M, N) = 0
440 NEXT N, M : GOTO 20
500 PRINT
510 PRINT " TWO CARDS REMAIN . . . DO YOU CONTINUE ? "
520 INPUT " YES (1), NO (2) ? " ; B
530 ON B GOTO 400, 550
540 GOTO 510
550 END
```



Points of the program:

Statement number 100 and 110	Turning up a new card.
Statement number 120	If a newly turned up card has been previously turned up, another card is to be turned up.
Statement number 130	Mark dealt cards with "–1".
Statement numbers from 420 to 440	All cards are collected and their marks are returned to "0".

Program Recording (SAVE)



It is convenient if the value of the counter is noted when using the **SAVE** command.

SAVE CR
key operation

SAVE "NAME" CR
key operation

The name should be no more than 16 characters.

TV screen display

RECORD · PLAY

Preparations for recording are O.K. !!

Cassette tape recorder operation

Press the **RECORD** and **PLAY** buttons

Recording start!!

TV screen display

WRITING NAME

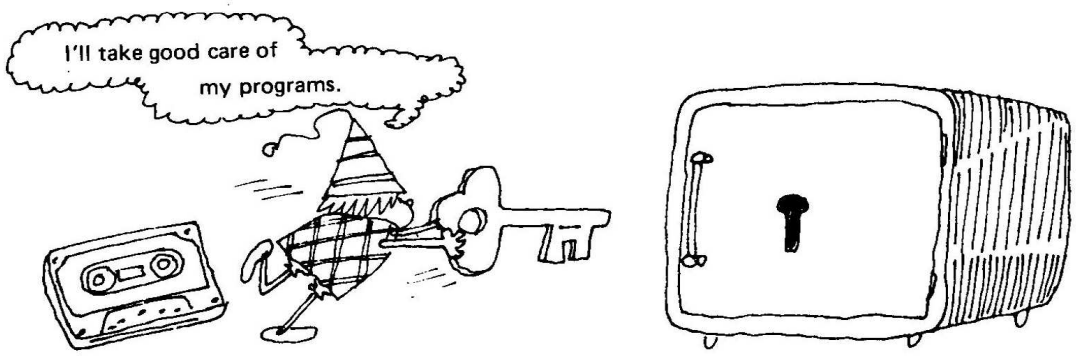
Recording in progress

No name is displayed with no file name given to a program.

TV screen display

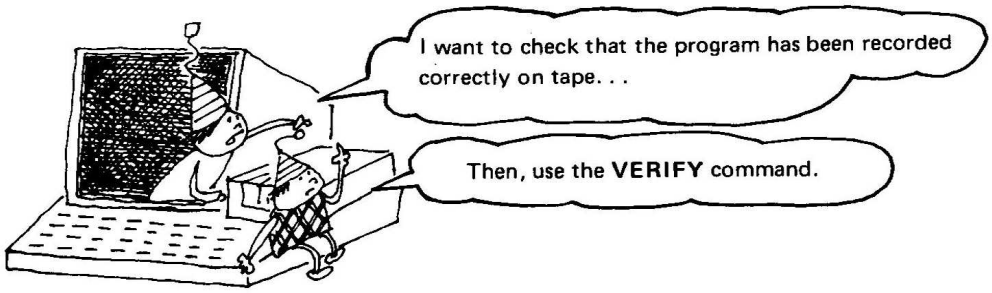
READY

Recording completed



Use of VERIFY and LOAD Commands

VERIFY



(1) Rewinding the tape.

(2) **VERIFY "NAME"** CR
"Name" is not required if a file name is not given.

(3) ↓ **PLAY**. . . TV screen display.

(4) **PLAY** button pressed. Cassette tape recorder operation.

(5) **FOUND** **** . . . . . This is on display if another program is found. Its file name, if given, is displayed as ****.
***** (file name)

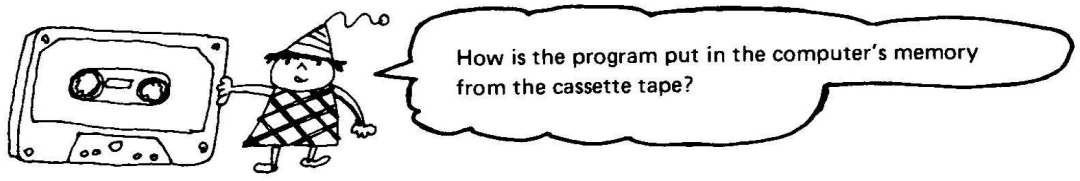
(6) **FOUND NAME** . . . . . Display indicating that the program to be verified was found.

(7) **VERIFYING NAME**
Checking that recording is correctly done is in progress.

(8) **DATA ERROR**
READY
meaning an error in recording. When this is on display, do **SAVE** once again.

(8) **OK**
READY
With this display, recording is verified.

LOAD



(1) **LOAD "NAME"** CR

(2) ↓ **PLAY**

(3) **Press the PLAY button.**

(4) **FOUND** ****

(5) **FOUND NAME**

Display and operation are same as the steps (3) to (6) of the **VERIFY** command, except for (4) and (5) which are the displays for the **LOAD** command.

(6) **LOADING NAME** ..... Transfer to the computer in progress (loading).

(7) **READY** ..... Loading completed.

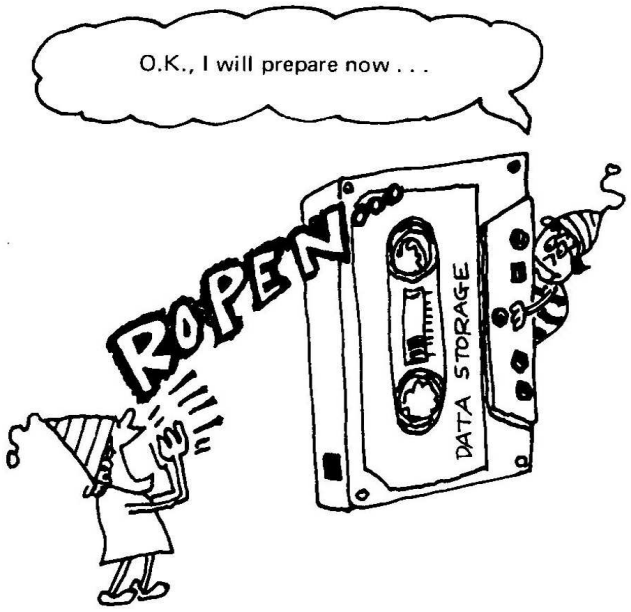
Data can also be Stored on Cassette Tape

Data storage is also required if programs can be stored. . . .
To do so, 5 more statements must be learned. Then, a cassette tape can be used as a storage of data.

WOPEN	This prepares for data writing. It also serves to name a group of data.
PRINT/T	Identical in use to the PRINT statement, this writes data on a cassette tape.
ROPEN	This statement prepares for data readouts. It serves to find a data group with the name given.
INPUT/T	Identical in use to the INPUT statement, this reads data out of the cassette tape.
CLOSE	This statement must be executed before ROPEN if WOPEN is executed or before WOPEN if ROPEN is executed.

To store data, numerals from 1 to 99 are first written on a casstte tape. The "DATA" at statement number 10 is the name given to a group of data to be written. A maximum of 16 characters can be used to name a group of data. Of course, it is unnecessary to have a name if so desired.

```
10 WOPEN "DATA"
20 FOR X=1 TO 99
30 PRINT/T X
40 NEXT X
50 CLOSE
60 END
```



Now, it is time to read the data which has just been written. First, rewind the cassette tape, then execute the following:

```
10 WOPEN "DATA"
20 FOR X=1 TO 99
30 INPUT/T A
40 PRINT A
50 NEXT X
60 CLOSE
70 END
```

Why not execute the above program again with 100 substituted for 99 at statement number 20? An error will occur this time. Because the 100th data was not originally written. It is impossible to memorize the written data counts. For this, a numeral, for example, −99999999 unrelated to that used for data is written as a mark at the end of written data.

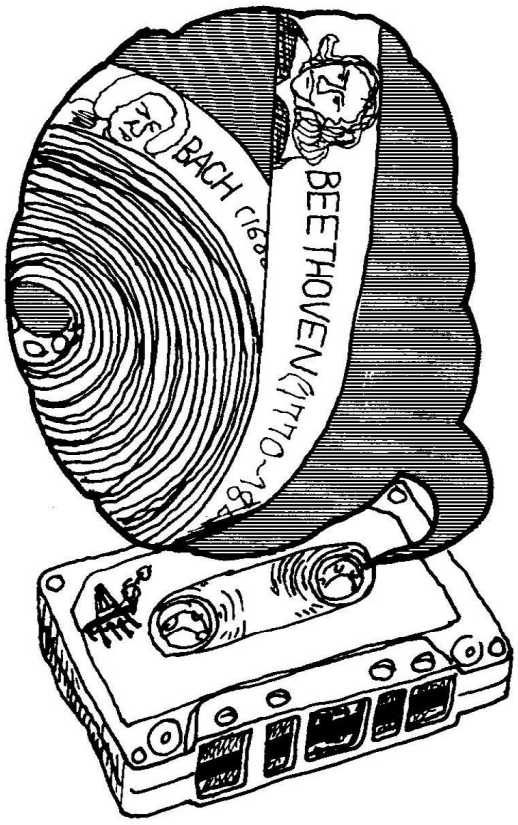
```
10 WOPEN "DATA"
20 FOR X=1 TO 99
30 PRINT/T X
40 NEXT X
50 PRINT/T −99999999
60 CLOSE
70 END
```

```
10 ROPEN "DATA"
20 FOR X=1 TO 200
30 INPUT/T A
40 IF A=−99999999 THEN 70
50 PRINT A
60 NEXT X
70 CLOSE
80 END
```

Technique to Memorize a Music History

Statements for data storage and readouts can also be used for strings.
The five composers' names are written on the cassette tape and read out of it.

```
10 DIM N$ (5)
20 N$ (1) = "BACH "
30 N$ (2) = " MOZART"
40 N$ (3) = " BEETHOVEN "
50 N$ (4) = " CHOPIN "
60 N$ (5) = " BRAHMS "
70 WOPEN " GREAT MUSICIANS "
80 FOR J = 1 TO 5
90 PRINT/T N$ (J)
100 NEXT J
110 CLOSE
120 END
```



This is identical to numeric data writing. Then, readouts are done as follows:

```
200 DIM M$ (5)
210 ROPEN " GREAT MUSICIANS "
220 FOR K = 1 TO 5
230 INPUT/T M$ (K)
240 PRINT M$ (K)
250 NEXT K
260 CLOSE
```

With this, writing and readout are completed. As you may have noticed, the name of string variable N$ used for writing is different from that of string variable M$ used for readouts. Since the value itself is written in the cassette tape as data, it has nothing to do with the name of the substituted variable. This makes it possible to change the variable name in the program as long as the string data is read by the string variable and the numeral data by the numeral variable.
Now, from what you have learnt so far, let's generate a data file with mixed numeric and string data. To also write the years when the previous composers died, for example, the following statements should be modified from the previous program.

```
15 DIM D (5)
65 D (1) = 1750 : D (2) = 1791 : D (3) = 1827
67 D (4) = 1849 : D (5) = 1897
90 PRINT/T N$ (J), D (J)
```

It is clear from the above that the generated file stores string and numeric data in alternate sequence. Accordingly, the readouts of the file must match the alternate sequence, for which statement numbers 200, 230 and 240 should be modified as follows:

```
200 DIM M$ (5), T (5)
230 INPUT/T M$ (K), T (K)
240 PRINT M$ (K), T (K)
```

With those statements remaining unmodified, the numeric data is transferred to the string variable M$ ( ), causing an error to occur.

List of School Work Results Prepared by a Smart Teacher

This is a program for recording the results of French, English and science for a certain class.

```
10 INPUT " HOW MANY STUDENTS IN THE CLASS ? " ; N
20 DIM N$ (N), K (N), E (N)
30 DIM R (N)
40 A$ = " (MARKS) "
50 FOR X = 1 TO N
60 PRINT : PRINT X
70 INPUT " NAME PLEASE ? " ; N$ (X)
80 PRINT " FRENCH " ; A$ ; : INPUT K (X)
90 PRINT " ENGLISH " ; A$ ; : INPUT E (X)
100 PRINT " SCIENCE " ; A$ ; : INPUT R (X)
120 NEXT X
130 WOPEN " RESULT "          <---- For writing a data group named "RESULT".
140 PRINT/T N                 <---- Writing the number of students in the class.
150 FOR X = 1 TO N
160 PRINT/T N$ (X), K (X), E (X), R (X)   <---- Writing the marks for students.
170 NEXT X
180 CLOSE                     <---- Writing completed.
```

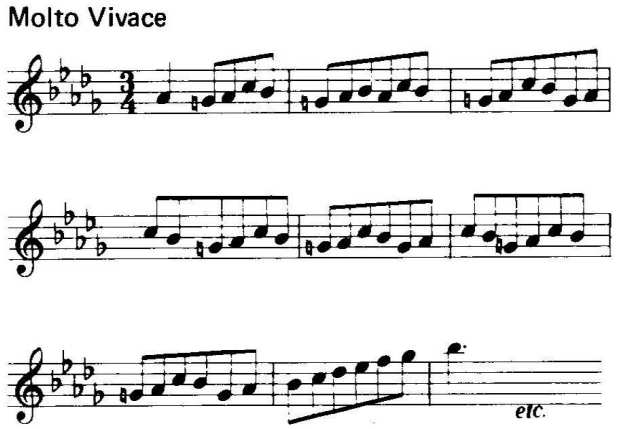
Now, let's read the written data of results, and calculate the mean of individual students' points and the mean of each subject.

```
10 ROPEN " RESULT "           <---- For finding the data group named " RESULT ".
20 INPUT/T N                  <---- Readouts of the number of students in a class.
30 DIM N$ (N), K (N), E (N)
40 DIM R (N)
50 FOR X = 1 TO N
60 INPUT/T N$ (X), K (X)      <---- Readouts of the name and marks for French.
70 INPUT/T E (X), R (X)       <---- Readouts of marks for English and science.
80 NEXT X
90 CLOSE                      <---- Readouts completed.
100 PRINT TAB (12) ; " FRENCH " ;
110 PRINT TAB (19) ; " ENGLISH " ;
120 PRINT TAB (27) ; " SCIENCE " ;
130 PRINT TAB (34) ; " MEAN "
140 FOR X = 1 TO N
150 PRINT N$ (X) ; TAB (11) ; K (X) ;
160 PRINT TAB (18) ; E (X) ;
170 PRINT TAB (26) ; R (X) ;
190 PRINT TAB (33) ; INT (10/3 * (K (X) + E (X) + R (X))) / 10
200 K (0) = K (0) + K (X) : E (0) = E (0) + E (X)
210 R (0) = R (0) + R (X)
220 NEXT X : PRINT " MEAN " ;
230 PRINT TAB (11) ; INT (10 * (K (0) / N)) / 10 ;
240 PRINT TAB (18) ; INT (10 * (E (0) / N)) / 10 ;
250 PRINT TAB (26) ; INT (10 * (R (0) / N)) / 10
260 END
```

Music Library Kept on Tapes

This data file is indispensable to generate a "Music Library" as discussed in the paragraph "MUSIC Statement".
Data for tunes is string data consisting of various symbol groups. If a data group is named per tune, any tune can be picked out of those recorded on the tape when its name is designated.

For example, a tune can be picked up from this music library for use in the music box of your timer, with some modifications. The tunes in the music library can also be used for programs of games and graphics, providing a number of applications.



F. Chopin " Puppy Waltz "

To write the etude of F. Kroepsch used on page 87 into a date file, the following changes mus be made:

```
300 WOPEN " ETUDE "
310 PRINT/T J1$, J2$, J3$, N1$, N2$, N3$
320 CLOSE
```

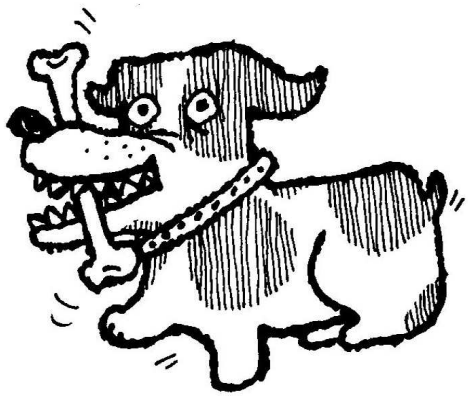
Attention is required to the fact that the character count for data writing should be within 255 characters. If written as follow;

```
305 MA$ = J1$ + J2$ + J3$ : MB$ = N1$ +N2$ +N3$
310 PRINT/T MA$, MB$
```

the contents of string variables MA$ and MB$ exceed 255 characters, which make data writing incomplete.

The length of string data may vary with each tune, therefore it is neccessary to write data indicating the end of each tune so that a data error does not occur in data readouts. End mark of tune " ◧■ ", for example, makes each tune consist of string data within 100 characters for execution given below:

```
500 ROPEN " PUPPY WALTZ "
510 FOR A = 1 TO 100
520 INPUT/T M$ (A)
530 IF M$ (A) = "◧■" THEN 550
540 NEXT A
550 CLOSE
```

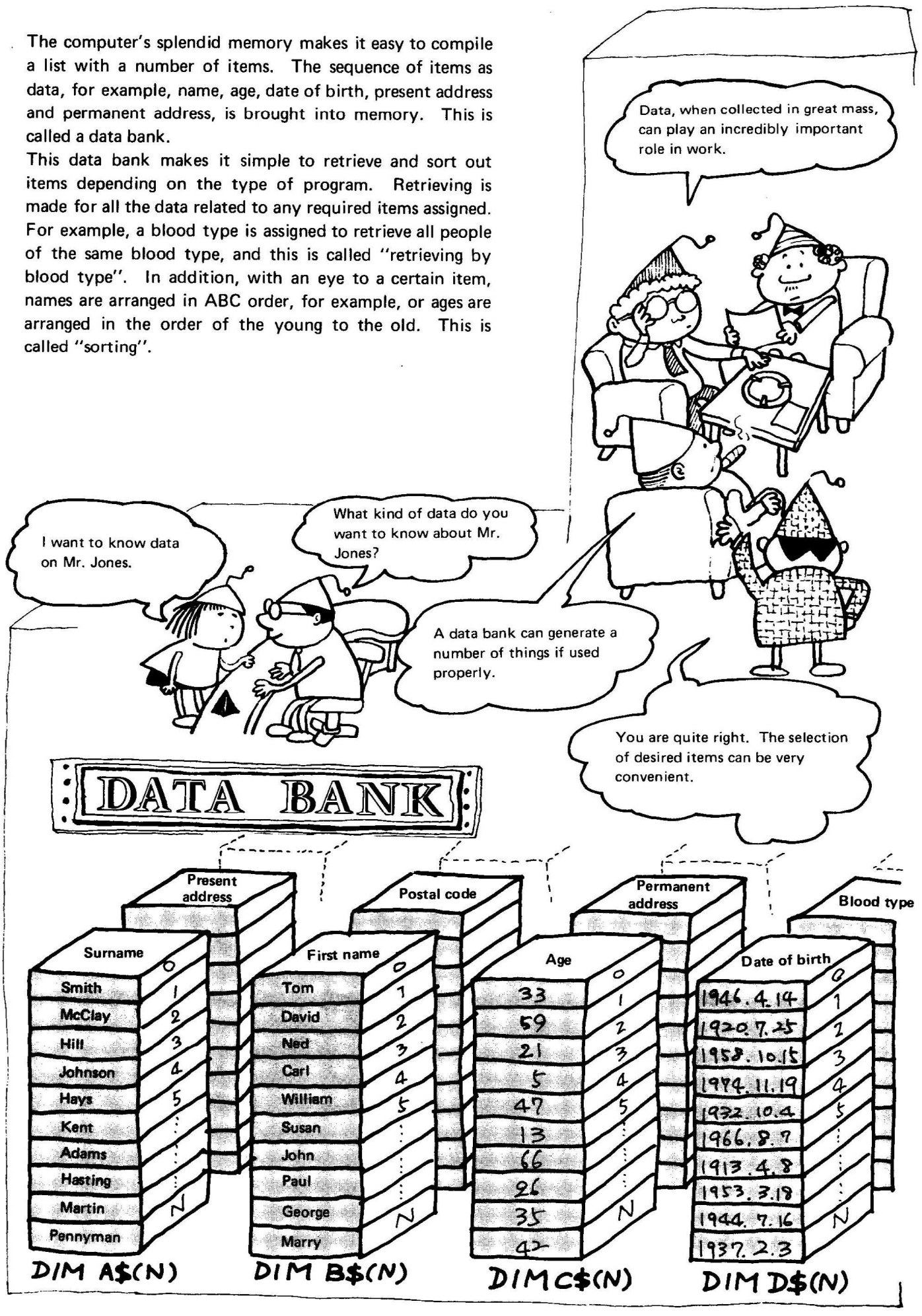


This can read the "Puppy Waltz" completely.

Data Bank is a Computer's Speciality

The computer's splendid memory makes it easy to compile a list with a number of items. The sequence of items as data, for example, name, age, date of birth, present address and permanent address, is brought into memory. This is called a data bank.

This data bank makes it simple to retrieve and sort out items depending on the type of program. Retrieving is made for all the data related to any required items assigned. For example, a blood type is assigned to retrieve all people of the same blood type, and this is called "retrieving by blood type". In addition, with an eye to a certain item, names are arranged in ABC order, for example, or ages are arranged in the order of the young to the old. This is called "sorting".



Telephone Number List is also a Data Bank

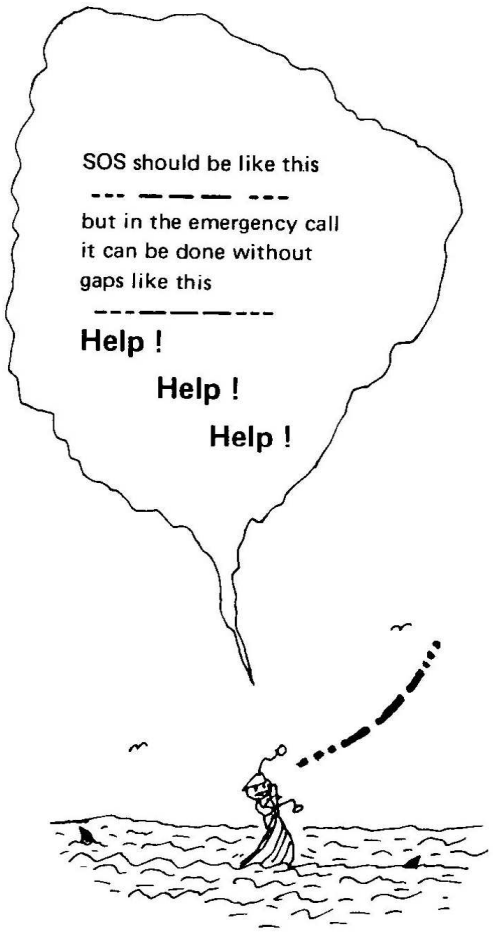
With the above understood, a summary is made of the program in which string data is put into the memory of the DATA statement. Based on this program, modifications are possible so that the address and postal code are also avilable.

```
10 N = 12
20 DIM M$ (N) ← Surname
30 DIM N$ (N) ← First name
40 DIM A$ (N) ← Home dialling code
50 DIM B$ (N) ← Home telephone number
60 DIM C$ (N) ← Work dialling code
70 DIM D$ (N) ← Work telephone number
80 DIM F (N)
90 FOR K = 1 TO N
100 READ M$ (K), N$ (K)
110 READ A$ (K), B$ (K) ◀ Data readouts
120 READ C$ (K), D$ (K)
130 NEXT K
140 PRINT : PRINT : X = 0
150 PRINT "WHAT IS THE SURNAME " ;
160 INPUT X$                    ← Key-in the name to be retrieved.
170 FOR K = 1 TO N
180 IF M$ (K) = X$ THEN X = X + 1 : F (X) = K ← Retrieving by use of the surname.
190 NEXT K
200 IF X <> 0 THEN 240
210 PRINT " NO RELEVANT PERSON FOUND ! "
220 PRINT " PLEASE RE – ENTER "
230 GOTO 140
240 PRINT : PRINT
250 FOR K = 1 TO X
260 L = F (K)                                   ← For display of persons with the same surname.
270 PRINT " NAME " ; TAB (11) ; ": " ; N$ (L) ; "  " ; M$ (L)
280 PRINT " HOME NUMBER :  " ;
290 PRINT " (" ; A$ (L) ; ")  " ; B$ (L)
300 PRINT " WORK NUMBER:  " ;
310 PRINT " (" ; C$ (L) ; ")  " ; D$ (L): PRINT
320 NEXT K
330 GOTO 140
340 DATA JONES, JOHN, 01, 364 9617, 01, 969 3678
350 DATA DAVIS, PETER, 021, 396 2137, 01, 323 6146
360 DATA SMITH, PAUL, 0449, 73246, 0449, 71277
370 DATA JONES, DAVID, 061, 631 1235, 061, 312 1975
380 DATA RICHARDS, ROBIN, 0273, 61976, 0903, 47216
390 DATA SMITH, HARRY, 01, 638 2174, 29, 147636
400 DATA LAKE, COLIN, 4967, 13642, 4967, 32132
410 DATA WATSON, JOHN, 01, 961 2431, 0427, 21369
420 DATA CARTER, DAVID, 6317, 21974, 01, 316 2638
430 DATA HOMLES, FRANK, 2238, 76194, 2238, 78352
440 DATA JONES, FRED, 9743, 61665, 01, 424 6913
450 DATA WILSON, JAMES, 01, 692 5687, 0374, 68421
460 END
```

SOS in Morse Code

The Morse code was invented by Samuel F. Breese Morse, an American artist, in 1838, and is one of the most important communications media even today. The principle is simple. It sets up the ratio of times when a specified wave of frequency is produced and not produced.

Prolonged sound ———	Transmission of sound 3 times as long as short sound.
Short sound —	
Pause	No sound for the same period of time as short sound



The Morse code is based on the combination of these three sounds to represent the necessary symbols. Shown below is part of the Morse code, according to which try to strike SOS. Very difficult ? The Morse code requires practice until your fingers move naturally and quickly without thinking of where to press.

To make this easy, the program for the Morse code is generated in the following section. Statement numbers 20 to 270 are strings to generate signals from A to Z, and statement numbers 290 to 380 for those from 0 to 9. Brief description is given of the program.

□A5 ——→ Sound A (la) in the high frequency range with its tonal length of 5 (equivalent to the prolonged signal of the Morse code).

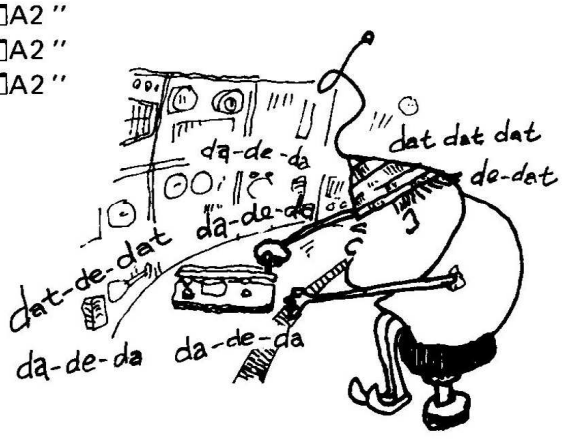
□A2 ——→ Sound A (la) in the high frequency range with its tonal length of 2 (equivalent to the short signal of the Morse code).

R2 ——→ Pause with no sound with its length of 2.

A	·—	G	——·	N	—·	T	—	Z	——··	6	—····
B	—···	H	····	O	———	U	··—	1	·————	7	——···
C	—·—·	J	·———	P	·——·	V	···—	2	··———	8	———··
D	—··	K	—·—	Q	——·—	W	·——	3	···——	9	————·
E	·	L	·—··	R	·—·	X	—··—	4	····—	0	—————
F	··—·	M	——	S	···	Y	—·——	5	·····	1	·—

Signals in Dots and Dashes

```
10 DIM A1 (100), M$ (127)
20 M$ (65) = "□A2R2□A5 "
30 M$ (66) = "□A5R2□A2R2□A2R2□A2 "
40 M$ (67) = "□A5R2□A2R2□A5R2□A2 "
50 M$ (68) = "□A5R2□A2R2□A2 "
60 M$ (69) = "□A2 "
70 M$ (70) = "□A2R2□A2R2□A5R2□A2"
80 M$ (71) = "□A5R2□A5R2□A2 "
90 M$ (72) = "□A2R2□A2R2□A2R2□A2 "
100 M$ (73) = "□A2R2□A2 "
110 M$ (74) = "□A2R2□A5R2□A5R2□A5 "
120 M$ (75) = "□A5R2□A2R2□A5"
130 M$ (76) = "□A2R2□A5R2□ A2R2□A2 "
140 M$ (77) = "□A5R2□A5 "
150 M$ (78) = "□A5R2□A2 "
160 M$ (79) = "□A5R2□A5R2□A5 "
170 M$ (80) = "□A2R2□A5R2□A5R2□A2 "
180 M$ (81) = "□A5R2□A5R2□A2R2□A5 "
190 M$ (82) = "□A2R2□A5R2□A2 "
200 M$ (83) = "□A2R2□A2R2□A2 "
210 M$ (84) = "□A5 "
220 M$ (85) = "□A2R2□A2R2□A5 "
230 M$ (86) = "□A2R2□A2R2□A2R2□A5 "
240 M$ (87) = "□A2R2□A5R2□A5 "
250 M$ (88) = "□A5R2□A2R2□A2R2□A5 "
260 M$ (89) = "□A5R2□A2R2□A5R2□A5 "
270 M$ (90) = "□A5R2□A5R2□A2R2□A2 "
280 REM NO.
290 M$ (48) = "□A5R2□A5R2□A5R2□A5R2□A5 "
300 M$ (49) = "□A2R2□A5R2□A5R2□A5R2□A5 "
310 M$ (50) = "□A2R2□A2R2□A5R2□A5R2□A5 "
320 M$ (51) = "□A2R2□A2R2□A2R2□A5R2□A5 "
330 M$ (52) = "□A2R2□A2R2□A2R2□A2R2□A5 "
340 M$ (53) = "□A2R2□A2R2□A2R2□A2R2□A2 "
350 M$ (54) = "□A5R2□A2R2□A2R2□A2R2□A2 "
360 M$ (55) = "□A5R2□A5R2□A2R2□A2R2□A2 "
370 M$ (56) = "□A5R2□A5R2□A5R2□A2R2□A2 "
380 M$ (57) = "□A5R2□A5R2□A5R2□A5R2□A2 "
390 REM " SPACE "
400 M$ (32) = " R5"
1000 INPUT " TYPE IN A MASSAGE " ; A$
1010 FOR J = 1 TO LEN (A$)
1020 A1 (J) = ASC (MID$ (A$, J, 1))
1030 NEXT J
1040 FOR J = 1 TO LEN (A$)
1050 MUSIC M$ (A1 (J)), " R5 "
1060 NEXT J
1070 GOTO 1000
```



Key in alphabet from A to Z and munerals from 0 to 9. For example, when you key-in "I LOVE YOU", the Morse code will be generated accordingly. Using the Morse code, you can declare your love to your sweetheart!

Unending "Time".....

At the end of this guide to the BASIC Language, the program for the "Perpetual Calendar" is introduced. It requires no detailed explanation. Our "time" continues eternally.

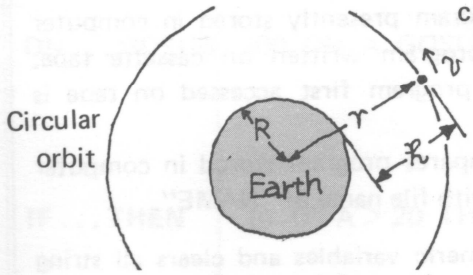
```
5 DIM M$ (12), W$ (7)
10 FOR K = 1 TO 12 : READ M$ (K) : NEXT K
20 FOR K = 1 TO 7 : READ W$ (K) : NEXT K
30 INPUT "YEAR PLEASE ? " ; Y : INPUT "MONTH PLEASE ? " ; MT
40 H = MT : GOSUB 400 : K2 = YB + 1
50 H = MT + 1 : GOSUB 400 : K1 = YB + 1
60 N = K1 – K2 : IF N >= 0 THEN L = 28 + N : GOTO 70
65 L = 35 + N
70 IF MT = 12 THEN L = 31
75 PRINT "■" : GOSUB 190
80 PRINT TAB (8) ; Y ; "  " ; M$ (MT) : PRINT : T = 2
90 FOR N = 1 TO 7 : PRINT TAB (T) ; W$ (N) ; : T = T + 4 : NEXT N : PRINT
100 T = 0 : IF K2 = 0 THEN 120
110 FOR N = 1 TO K2 : PRINT TAB (T) ; : T = T + 4 : NEXT N : T = T – 4
120 FOR N = 1 TO L : N$ = STR$ (N) : J = LEN (N$)
130 PRINT TAB (T + 5–J) ; N$ ; : T = T + 4
140 IF T = 28 THEN T = 0 : PRINT
150 NEXT N
160 IF T <> 0 THEN PRINT
170 GOSUB 190
180 PRINT "■" : GOTO 30
190 FOR Z = 1 TO 31 : PRINT "*" ; : NEXT Z : PRINT : RETURN
200 DATA JAN, FEB, MAR, APR, MAY, JUN
210 DATA JUL, AUG, SEP, OCT, NOV, DEC
220 DATA SUN, MON, TUE, WED, THU, FRI, SAT
230 END
400 X = Y
410 N = H–3 : J = 12 : GOSUB 600 : MM = Z
420 IF MM > 9 THEN X = X – 1
430 N = X : J = 400 : GOSUB 600 : X = Z
440 X4 = INT (X/4) : X1 = INT (X/100)
450 KY = X + X4 – X1
460 N = MM : J = 5 : GOSUB 600 : MZ = Z
470 M5 = INT (MM/5) : M2 = INT (MZ/2)
480 N = MZ : J = 2 : GOSUB 600 : P = Z
490 KM = 13 * M5 + 5 * M2 + 3 * P
500 N = KY + KM + 3 : J = 7 : GOSUB 600 : YB = Z
510 RETURN
600 REM Z = N, J
610 K = INT (N/J)
620 Z = N – K * J
630 IF Z < 0 THEN Z = Z + J
640 RETURN
```


```
**********************************
          1980  JAN
SUN MON TUE WED THU FRI SAT
                1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30  31
**********************************

YEAR PLEASE ■
```

Miniature Space Dictionary

If you are interested in space, including astronomy and man-made satellites, you might like to try calculations and graphic drawings by using the computer. Shown below are equations and values required for such attempts. Unlike the earth, the movement of objects in space should match mathematical calculations without any complexity caused by atmospheric resistance. For more accurate values, however, consideration must be given to the effects by the planets, the perturbation caused by strains in the form of the earth and gas pressure in space, even though rarefied. There is air of 10^{-9} mmHg at an altitude of 800km in space, for example. In addition, a man-made satellite stationed at an altitude of approx. 36,000km tilts approximately 1 degree per year in its orbit being affected by other heavenly bodies.

Motion velocity of a satellite in circular orbit : $v = \sqrt{\frac{G \cdot M}{r}}$ (m/s)



Period : $T = 2\pi\sqrt{\frac{r^3}{G \cdot M}}$ (s)

$G = 6.67 \times 10^{-11}$ $(N \cdot m^2/kg^2)$

Universal gravitation constant

M : Mass of the earth (kg)

r : Distance from the earth's center to satellite (m)

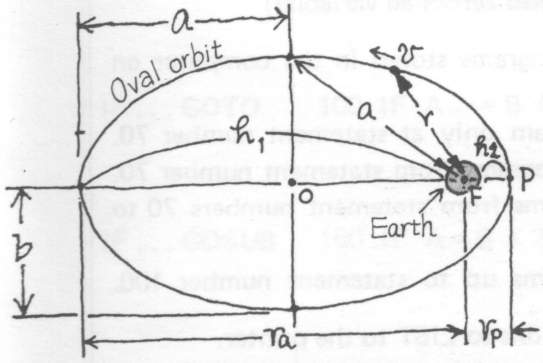
R: Radius of the earth

h: Height from the earth surface

Motion velocity of a satellite in oval orbit : $v = \sqrt{G \cdot M\left(\frac{2}{r} - \frac{1}{a}\right)}$ (m/s)

Period : $T = 2\pi\sqrt{\frac{a^3}{G \cdot M}}$ (s)

Long radius : $a = \frac{h_1 + h_2 + 2R}{2}$ (m)



Short radius : $b = \sqrt{a^2 - (a \cdot e)^2}$ (m)

Eccentricity : $e = OF/OP = \sqrt{\frac{a^2 - b^2}{a}}$

$r_p = a(1 - e)$ (m)

$r_a = a(1 + e)$ (m)

	Mass (1 for the Sun)	Equatorial radius	Eccentricity	Averaged distance from the Sun (a)
Sun	1.000	696 000km	–	–
Mercury	0.166×10^{-6}	2 440	0.20563	0.57910×10^{8}km
Venus	2.448×10^{-6}	6 056	0.00678	1.08210 "
Earth	30.034×10^{-7}	6 378	0.01672	1.49600 "
Mars	3.227×10^{-7}	3 390	0.09338	2.27944 "
Jupiter	95.479×10^{-5}	71 400	0.04829	7.7834 "
Saturn	2.856×10^{-4}	60 400	0.05604	14.2700 "
Uranus	4.373×10^{-5}	23 700	0.04613	28.7103 "
Neptune	5.178×10^{-5}	25 110	0.01004	44.971 "
Pluto	0.552×10^{-6}	3 400	0.24842	59.136 "
Moon	3.694×10^{-8}	1 738	0.0549 *	384 400 km *

Mass of the Sun = 1.991×10^{30} kg

* Value to the earth

Source: Chronological Table of Science

Summary of the BASIC Instructions

Direct mode command

LOAD	LOAD	Reads BASIC program stored on cassette tape. Since no file name is given, in this case, the program first found is read.
	LOAD "NAME" ← File name	Reads BASIC program with file name of "NAME". (File name is valid up to 16 characters.)
SAVE	SAVE	Writes program presently stored in computer to cassette tape.
	SAVE "NAME" ← File name	Similar to the above, writes program with file name of "NAME".
VERIFY	VERIFY	Compares program presently stored in computer and BASIC program written on cassette tape. In this case, program first accessed on tape is compared.
	VERIFY "NAME" ← File name	Similarly, compares program stored in computer and program with file name of "NAME".
CLR	CLR 10 CLR	Zeroes all numeric variables and clears all string variables. This can be used during program execution.
RUN	RUN	Executes program. In this case, however, there's no statement number and the smallest statement number is first executed.
	RUN 100	Execution begins at statement number 100. If an unused statement number is assigned an error occurs. (RUN also zeroes all variables)
LIST	LIST	Displays all programs stored in the computer on the TV screen.
	LIST 70	Displays program only at statement number 70.
	LIST 70 –	Displays all programs from statement number 70.
	LIST 70 – 100	Display programs from statement numbers 70 to 100.
	LIST – 100	Display programs up to statement number 100.
LIST/P	LIST/P LIST/P 70 LIST/P 70 – 100 LIST/P –100	Performs functions to LIST to the printer. (This causes an error in a computer configuration without a printer.)
NEW	NEW 10 NEW	Clears program presently stored in computer, and make all variables 0 or space. This can also be used during program execution.
CONT	CONT	When program execution is stopped (with BREAK key or STOP and END statements during program execution), this command restarts execution from the stop position. (If program is edited using statement numbers, this command cannot be used. The direct mode command can be used.)
BYE	BYE 10 BYE	Stops the use of BASIC instructions, and returns to monitor control. This can be used during program execution. Use with caution !!

Statements

GOTO	100 GOTO 200	Jumps from statement numbers 100 to 200.
GOSUB	100 GOSUB 700	Branches to subroutine at statement number 700.
RETURN	800 RETURN	Ends subroutine execution and returns to next statement number assigned by GOSUB statement in the main program.
ON . . . GOTO	10 ON A GOTO 70, 80, 90	Jumps to statement number 70 if value of variable A is 1, statement number 80 if value is 2, and statement number 90 if the value is 3. Executes next statement if A = 0 or A ≧ 4. ON includes INT, and if A is 2.7, A is regarded as 2 jumping to statement number 80.
ON . . . GOSUB	100 ON A GOSUB 700, 800	Branches to statement number 700 if value of variable A is 1, and statement number 800 if value is 2. Executes next statement if A = 0 or A ≧ 3. ON includes INT.
IF . . . THEN	10 IF A > 20 THEN 100	Jumps to statement number 100 if A is greater than 20. Executes next statement if A is less than 20.
	20 IF A <> 10 THEN 200	Jumps to statement number 200 if variable A is not equal to 10. Executes next statement if A equals 10.
	30 IF B < 3 THEN B = B + 3	Substitutes B + 3 value for variable B if value of variable B is less than 3. Executes next statement if B is greater than 3.
	40 IF B > 7 THEN PRINT B	Displays B value on TV screen if value of variable B is greater than 7. Executes next statement if B is less than 7.
IF . . . GOTO	100 IF A >= B GOTO 10	Jumps to statement number 10 if value of variable A is greater than that of variable B. Executes next statement if A is less than B.
IF . . . GOSUB	100 IF A = B * 2 GOSUB 700	Branches to statement number 700 if the value of variable A is double that of variable B. Executes next statement if not. (If multi-statement follows conditional sentence, ON statement is executed when condition is not met, while IF statement shifts to next statement number when condition is not met, and multi-statement is omitted.)
FOR . . . NEXT STEP	10 FOR A = 1 TO 10 20 PRINT A 30 NEXT A	Statement number 10 states that variable A should be changed from 1 to 10, and the first value of A is 1. Statement number 20 states value of A is to be displayed on TV sereen, therefore 1 is on display. At statement number 30, the value of A is 2, when this loop is repeated. Accordingly, 2 is displayed as the value of A. Thus, this loop is repeated until the value of A becomes 10. (When the loop is ended, the value of A is 11.)

FOR . . NEXT STEP

```
10 FOR B =2 TO 8 STEP 3
20 PRINT B↑2
30 NEXT
```

Statement number 10 states that value of B should be increased from 2 to 8 by increments of 3. With the value of STEP made minus, the value of the variable can be decreased. At statement number 20, B squared is displayed. Statement number 30 is not NEXT B. In this case, however, when no variable is given to NEXT statement, it is combined with the nearest FOR loop. In example at left, FOR . . . NEXT loop about variable B is executed. It is suggested that the variable name is given to the NEXT statement.

```
10 FOR A = 1 TO 3 ──────┐
20 FOR B = 10 TO 30 ──┐ │
30 PRINT A, B  Loop B │ │
40 NEXT B ────────────┘ │
50 NEXT A ──────────────┘
                Loop A
```

This is an example of double FOR . . . NEXT loops for variables A and B. Pay attention to the fact that B loop is placed inside A loop. Although double, triple, etc. loops are possible, inner loops must be enclosed by outer loops. The configuration of multi-loop is called "nesting".
In example at left, variable A is first set to 1 and variable B to 10. With A remaining at 1, B is changed from 10 to 11, 12, until it reaches 30 under execution of loop B. At statement number 50, value of A becomes 2 with loop B execution started again.

```
60 NEXT B : NEXT A
60 NEXT B, A
70 NEXT A, B
```

In substitution for NEXT statements at statement numbers 40 and 50, two statements at number 60 shown at left can be used. However, statement number 70 cannot be used, causing an error to occur.

READ . . DATA

Substitutes constant or string placed in DATA statement for variable shown in READ statement. First READ statement reads the first value of DATA statement, and second READ statement reads the second value of DATA statement. Variable in READ statement and corresponding value in DATA statement should match the form of variable, with numeral for numeric variable and string for string variable. In addition, no equation can be placed in DATA statement. DATA statement can be placed anywhere in the program, but at the beginning or end for convenience.

```
10 READ A, B, C, D
20 DATA 25, –0.5, 0, 500
```

In READ and DATA statements at left, values of 25, –0.5, 0 and 500 are substituted for variables A, B, C and D.

```
10 READ H$, H, S$, S, D$, D
20 DATA HEART, 3
30 DATA CLUB, 9
40 DATA SPADE, 6
```

In example at left, first value in DATA statement or string "HEART" is substituted for first variable in READ statement or string variable H$. Value 3 is substituted for second variable H, and thus read one after another.

```
10 READ X1$, X2$, Y$
20 DATA " A, B, C, D "
30 DATA "E, F, G", " : 16"
```

When string includes " , " and " : ", it is indicated by quotation marks. In example at left, strings in quotation marks of DATA statement are substituted for string variables X1$, X2$ and Y$, respectively.

RESTORE

```
10 READ A, B, C
20 RESTORE
30 READ D, E
40 DATA 3, 6, 9, 12, 15
```

In READ . . . DATA statements, values read from DATA statement are shifted as READ statement progresses. However, use of RESTORE statement allows readout value to return to the first DATA statement.
In example at left, values 3, 6 and 9 are substituted by READ statement at statement number 10 for variables A, B and C, respectively.
Since, however, RESTORE statement is placed next, values to be supstituted for variables D and E by READ statement at statement number 30 are not 12 and 15, but 3 and 6 are substituted respectively.

MUSIC
TEMPO

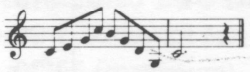
```
MUSIC "CDEFG"

300 TEMPO 7
310 MUSIC "DE#FGA"
```

This states music to be automatically played. With a tempo assigned by TEMPO statement, a string (equivalent to a group of notes assigned for scale and duration) for a melody between quotation marks " in MUSIC statement is converted to sound for delivery through a speaker.

Do, re, mi, fa and sol in the mid-range of the scale sound with quarter notes at TEMPO 4.
At statement number 300, TEMPO 7 is assigned (the fastest). At statement number 310, re, mi and fa with # sol and la in the mid-range are played at quarter note duration. With TEMPO statement not assigned, playing is conducted at TEMPO 4.

```
300 M1$ = "C3EG□C"
310 M2$ = "B3GD□G"
320 M3$ = "C8R5"
330 MUSIC M1$, M2$, M3$
```

In example at left, a melody is substituted for 3 string variables and MUSIC statement is executed. Melody at right on the scale is played. This has no TEMPO statement assigned, and TEMPO 4 is set for playing.



DEF FN

```
100 DEF FNA (X) = X ↑ 2 – X
110 DEF FNB (X) = LOG (X)
120 DEF FNC (Y) = LN (Y)
```

DEF FN defines functions. Statement 100 defines $X^2 - X$ as FNA (X), statement number 110, $\log_{10} X$ as FNB (X) and statement number 120, $\log_e Y$ as FNC (Y). Function is limited to 1 variable. Function names require the form of "variable ( ) " as illustrated A ( ), B ( ) and C ( ).

SET
RESET

```
SET X, Y
RESET X, Y
```

Lights the coordinate on TV screen as assigned by variables X and Y of SET statement. Lights assigned by RESET statement go out. The abscissa is from 0 to 79 starting at the left of TV sereen, while the ordinate is from 0 to 49 starting from the upper to lower part. This means top left corners are 0 and 0, and bottom right corners are 79 and 49. (Under normal operation, TV screen display consists of 40 for X and 25 for Y. Please note.)

```
SET 0,0
10 SET 79, 49
20 RESET 79, 49
30 GOTO 10
```

Lights the top left corner point.
Light at bottom right corner point repeatedly flashes at statement numbers 10 to 30.
X and Y assignments can be made by variable, equation or constant.

```
10 FOR J = 0 TO 49
20 SET J, J
```

Program at left is for drawing a slash from too left corner to bottom corner on the TV screen.

```
30 NEXT
100  FOR J=0 TO 79
110 SET J, SQR (J)
120 NEXT
```

At statement numbers 100 to 120, a parabola is drawn on the TV screen. SQR (J) does not always produce an integer, but the SET statement includes INT, making it possible to assign any coordinate. What requires attention is that when X and Y exceed 79 and 49, respectively, in the SET and RESET coordinates, X and Y turn to X – 80 and Y – 50, respectively. Coordinate assignment by minus value, or X and Y assigned by numerals exceeding 255 characters may cause an error to occur.

WOPEN

```
10 WOPEN
20 PRINT/T X, X$
```

Opens as a file for exclusive use with data to write contents of numeric variable and string variable onto cassette tape. (SAVE statement is available for program storage.)

```
30 WOPEN "COST"
40 PRINT/T X, X$
```

Similar to the above, this prepares for data of variable to be stored with file name of "COST". (16 characters are valid for file name.)

ROPEN

```
10 ROPEN
20 INPUT/T Y, Y$
```

Opens as a file for exclusive use with data for readouts of numeric variable and string variable stored on cassette tape. (LOAD command is available for readouts of programs.)

```
30 ROPEN "COST"
40 INPUT/T Y, Y$
```

Similar to the above, this prepares for readouts of variable data by retrieving a data file with file name of "COST".

CLOSE

```
50 CLOSE
```

This declares the end of using cassette tape for data a file. When WOPEN and ROPEN statements are used, the file must be closed.

INPUT

```
10 INPUT A
20 INPUT A$        <- String
30 INPUT "VALUE ? " ; A
```

Receives numeral for variable A from keyboard. Receives string for string variable A from keyboard. Before receiving keyin from keyboard, this displays string VALUE ?. Separating string from variable uses semicolon "; ". However, ? for numeral is not printed again. Numeric variable and string variable can be used in a mixed condition when separated by a comma " , ", but the from of variable should be followed when received.

```
40 INPUT A, A$, B, B$
```

INPUT/T

```
10 ROPEN
20 INPUT/T A
30 INPUT/T A$
40 INPUT/T A, A$, B, B$
```

Read function, identical to INPUT, from cassette tape. If, however, there is no data file opened by ROPEN statement, an error occurs.

GET

```
10 GET A
```

Receives numeral of 1 character for variable A from keyboard. With no key pressed in this case, 0 is received.

```
20 GET A$
```

Receives 1 string for string variable A$ from keyboard. With no key pressed in this case, A$ becomes a space.

PRINT

```
10 PRINT A
20 PRINT A$
30 PRINT A; A$, B ; B$
```

Displays value of variable A on TV screen.
Displays characters of string variable A on TV screen. Numeric and string variables can be used in a mixed condition. In addition, when a semicolon is used for separation, continuous display without spaces is possible. If a comma is used, display starts at next position (10 characters per separa-

	40 PRINT "COST = " ; A	tion). The contents of a string enclosed by quotation marks " are displayed as they are. In case of a PRINT statement alone, a line is fed.
	50 PRINT	
PRINT/T	10 PRINT/T A 20 PRINT/T A ; A$, B ; B$ 30 PRINT/T "COST = " ; A	Write function, identical to PRINT, to cassette tape. However, an error occurs unless there is data file opened by WOPEN statement.
PRINT/P	10 PRINT/P A 20 PRINT/P A ; A$, B ; B$ 30 PRINT/P "COST = " ; A	Executes function identical to PRINT to optional printer. An error occurs if no printer is provided. This statement can be used, irrespective of WOPEN statement.
SIZE	PRINT SIZE	Displays unused memory size (bytes) of the memory presently built into the computer.
TI$	TI$ = "102030" 10 TI$ = "102030"	Sets internal clock to 10 hours 20 minutes and 30 seconds AM. A number of 6 figures is used between quotation marks ".
	20 PRINT TI$	Displays the actual time of the internal clock.
TAB	10 PRINT TAB (X) ; A	Shifts cursor by X character space from the left hand on TV screen, and displays value of variable A.
	20 PRINT TAB (5) ; A$	Shifts cursor by 5 character space from the left hand on TV screen, and displays character train of string variable A$.
SPC	10 PRINT SPC (X) ; A 20 PRINT SPC (5) ; A$	Prints X space and displays value of variable A. Prints 5 spaces and displays character train of string variable A$.
DIM		When using variable with a subscript, the maximum of the subscript must be declared by this DIM (abbr. of dimension) statement. Subscript can be used with numerals from 0 to 255 (maximum). Subscript can also be declared by variable and equation in the above range.
	10 DIM A (20)	For variable A with subscript ( ), 21 arrays from A (0) to A (20) are prepared.
	20 DIM B (99, 99)	Variable B with double subscript ( ) can have 10000 arrays from B (0, 0) to B (99, 99) are prepared. Previous statement numbers 10 and 20 are combined for declaration in the form of statement number 30.
	30 DIM A (20), B (99, 99)	
	40 DIM C1$ (10)	For string variable C1$ with subscript ( ), 11 arrays from C1$ (0) to C1$ (10) are prepared.
	50 DIM AA (X), AB (X * 2)	For variable AA ( ) with subscript, arrays from AA (0) to AA (X) and for AB ( ), arrays from AB (0) to AB (X * 2) are prepared, respectively.
STOP	200 STOP	Stops program execution and waits for next instruction. With CONT command given, this continues program execution.
END	999 END	This ends program execution, but with CONT command given, it executes the next program.
REM	100 REM GAME	This is a comment and is omited during program execution. The colon " : " used even in REM statement is the separator of multi-statement, and 30 is substituted for variable A.
	200 REM GAME : A = 30	

PEEK	Decimal number ↓ 10 A = PEEK (24110) 20 B = PEEK (C) (With an address in BASIC program assigned, 32 is always read.)	Converts data in assigned address 24110 to a decimal number and substitutes it for variable A. Converts data in address assigned by variable C (regarded as decimal number) to a decimal number, and substitutes it for variable B. (Variables A and B are both set between 0 and 255.)
POKE	10 POKE A, D 20 POKE 24100, 32	Numerals (range of 0 to 255) indicated by variable D are stored in memory address assigned by variable A. Using POKE statement without care may cause the program to be destroyed. Therefore, don't use POKE statement when you are not sure about where to store it. In example, data 32 is stored in address 24100.
USR	10 USR (A) 20 USR (24150)	Shifts program control to memory address indicated by variable A. Fulfils the same function as subroutine call in machine language. This requires knowledge of machine language, and sufficient understanding needs to be obtained before actual use. Also, shifts program control to address 24150.
LIMIT	LIMIT A 10 LIMIT A LIMIT MAX 20 LIMIT MAX	Specifies memory limit usuable by BASIC program with variable A. When the above USR statement is used, it is convenient to secure a space region in the memory using this LIMIT statement and to store there the machine language program. (When the power is turned on, the memory limit is automatically set to the maximum memory.) Sets the memory limit to the maximum. When the memory limit is particularly set at the above statement number 10, memory can be returned to its maximum by using this statement.

String Processing Statements

LEFT$	300 A$ = LEFT$ (X$, N)	Substitutes the first to Nth character of string variable X$ for string variable A$. N can be either constant, variable or equation.
RIGHT$	600 B$ = RIGHT$ (X$, N)	Substitutes the last N characters of string variable X$ for string variable B$. N can be either constant, variable or equation.
MID$	900 C$ = MID$ (X$, M, N)	Substitutes the N characters from Mth character of string variable X$ for string variable C$. M and N can be either constant, variable or equation.
LEN	100 LX = LEN (X$) 110 LS = LEN (X$ + Y$)	Substitutes character length (No. of characters) of string variable X$ for variable LX. Substitutes the character length sum of string variables X$ and Y$ for variable LS.
ASC	200 A = ASC (X$)	Substitutes value of ASCII code (decimal), for the first character of string variable X$ for variable A.
CHR$	220 X1$ = CHR$ (A)	Contrary to ASC statement, this substitutes ASCII code character equal to value of variable A for string variable X1$. A can be either constant, variable or equation.

VAL	400 K = VAL (N$)	Substitutes numeric string of string variable N$ for variable K as value.
STR$	440 N$ = STR$ (K)	Contray to VAL statement, this substitutes numeral of variable K for string variable N$ as string.

Functions

ABS	100 A = ABS (X)	Substitutes absolute value IXI of variable X for variable A. Parenthesis can be either constant or equation. Example: ABS (−3) = 3 ABS (12) = 12
SGN	100 A = SGN (X)	Substitutes −1 if value of variable X is < 0, 0 if X = 0, and 1 if $X > 0$, for variable A. Parenthesis can either be constant or equation. Example: SGN (0.4) = 1 SGN (0) = 0 SGN (−400) = −1
INT	100 A = INT (X)	Determines maximum integer not exceeding X of value of variable X, and substitutes it for variable A. Parenthesis can either be constant or equation. Example: INT (3.87) = 3 INT (0. 6) = 0 INT (−3. 87) = −4
SIN	100 A = SIN (X)	Determines sin X value for value of variable X (radian) and substitutes it for variable A. Parenthesis can either be constant or equation. Since the relationship between the radian and degree is, $1 \text{ degree} = \frac{\pi}{180} \text{ radian}$
	110 A = SIN (30 ✳ π/180)	the substitution of sin 30° for variable A, for example, should be made as at statement number 110.
COS	200 A = COS (X)	Determines cosX value for value of variable X (radian), and substitutes it for variable A. Parenthesis can either be constant or equation. Calculations in degrees can be made in a similar manner to SIN function. Statement number 210 is to substitute value of cos200° for variable A.
	210 A = COS (200 ✳ π/180)	
TAN	300 A = TAN (X)	Determines tanX value for variable X value (radian) and substitutes it for variable A. Parenthesis can either be constant or equation. Calculations in degrees can be made in a similar manner to SIN function. Statement number 310 is a statement for substitution of tanY° value for variable A.
	310 A = TAN (Y ✳ π/180)	
ATN	400 A = ATN (X)	Determines $\tan^{-1}$ X value (radian) for variable X value and substitutes it for variable A. Parenthesis can either be constant or equation. Calculation result is the value between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Statement number 410 is a statement for conversion of $\tan^{-1}$X value to degrees and substitutes it for variable A.
	410 A = 180/π ✳ ATN (X)	

SQR	100 A = SQR (X)	Determines square root of X for variable X value and substitutes it for variable A. Parenthesis can either be constant or equation, but value must be plus or 0.
EXP	100 A = EXP (X)	Determines value of e^X (e to the Xth power) for variable X value and substitutes it for variable A. Parenthesis can either be constant to equation.
LOG	100 A = LOG (X)	Determines common logarithms $\log_{10}X$ value for variable X value and substitutes it for variable A. Parenthesis can either be constant or equation, but must be a plus value.
LN	100 A = LN (X)	Determines natural logarithms logeX value for variable X value and substitutes it for variable A. Parenthesis can either be constant or equation, but must be a plus value. To determine logarithms $\log_Y X$ with bottom of Y, statement number 110 or 120 can be used.
	110 A = LOG (X)/LOG (Y) 120 A = LN (X)/LN (Y)	
RND		This function generates random numbers with values from 0.00000001 to 0.99999999. There are two ways of processing, one with 0 or minus integer given in parenthesis and the other with plus integer given.
	100 A = RND (0) 110 B = RND (−3)	With 0 or minus integer given in parenthesis as at statement number 100 or 110, random number generation is initialized to generate a specified value at all times, and the same value is substituted for both A and B. With plus integer given in the parenthesis as at statement number 200 or 210, random number with value between 0.00000001 and 0.99999999 is generated, whenever RND function is used. In this case, however, this has nothing to do with the value of plus integer given in the parenthesis.
	200 A = RND (1) 210 B = RND (10)	

Arithmetic Operators

Numeral white on black base at left shows calculation priority order. Calculations in parentheses are further given priority.

	=	10 A = X + 3 (Substitution) 20 B = π	Substitutes 3+ variable X value for variable A. Substitutes π (3.1415927) value for variable B.
❶	↑	10 A = X ↑ Y (Power)	Substitutes calculation result of X^Y for variable A., (However, if X ↑ Y with X of minus value and Y of no integer, an error occurs.)
❷	−	10 A = −B (Minus sign)	0 −B is a subtraction, but "−" of −B is a minus sign, so please note.
❸	✳	10 A = X ✳ Y (Multiplication)	Substitutes product of X and Y values for variable A.
❸	/	10 A = X/Y (Division)	Substitutes quotient of X and Y values for variable A.
❹	+	10 A = X + Y (Addition)	Substitutes total of X and Y values for variable.
❹	−	10 A = X − Y (Subtraction)	Substitutes remainder of X and Y values for variable A.

Logical Operators

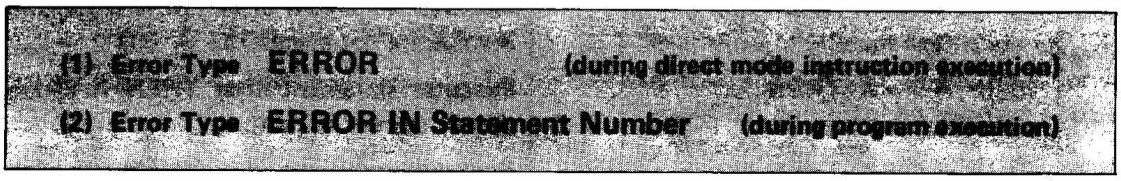
=	10 IF A = X THEN . . .	Executes statement after THEN if variable A and X value are equal.
	20 IF A$ = "XYZ" THEN . . . (Equal)	Executes statement after THEN if the contents of string variable A equal string XYZ.
<> or ><	10 IF A <> X THEN . . . (Not equal)	Executes statement after THEN if variable A and X value are not equal.
>= or =>	10 IF A >= X THEN . . . (Greater than or equal)	Executes statement after THEN if variable A is greater than or equal to X.
<= or =<	10 IF A <= X THEN . . . (Less than or equal)	Executes statement after THEN if variable A is less than or equal to X.
*	10 IF (A > X) * (B > Y) THEN . . . . . (Logical multiply)	Executes statement after THEN if variable A is greater than X and variable B is greater than Y.
+	10 IF (A > X) + (B > Y) THEN . . . . (Logical add)	Executes statement after THEN if variable A is greater than X or variable B is greater than Y.

Other Symbols

?	200 ? "A + B = " ; A + B 210 PRINT "A + B = " ; A + B	This can be used in substitution for PRINT statement. Therefore, statement numbers 200 and 210 are identical.
:	220 A = X : B = X ↑ 2 : PRINT A, B	Symbol representing pause in statement and is used for multi-statement. The multi-statement at statement number 220 includes 3 statements.
;	230 PRINT "AB" ; "CD" ; "E"	Executes PRINT statement continuously. At statement number 230, "ABCDE" is displayed on TV screen without space.
	240 INPUT "X = " ; X$	Displays "X=" on TV screen, and waits for string variable X$ to be keyed - in.
,	250 PRINT "AB", "CE", "E"	Executes PRINT statement with tabulation. At statement number 250, this displays AB first on TV screen, then CD at the position 10 characters to the right from A and finally E at the position 10 characters to the right from C.
	260 DIM A (20), B (30)	Example in which the comma is used for variable's separation.
"	270 A$ = "SHARP BASIC" 280 PRINT "*" ; A$; "*"	Indicates that a string is between quotation marke " ".
$	290 A1$ = LEFT$ (A$, 5) 300 A2$ = "MZ – 80 K"	Indicates string variable.
π	400 S = SIN (X * π/180)	Circular constant 3.1415927 is represented by π.

Error Messages

In case an error occurs during computer operation, an error message is sent from the computer to the TV screen in the following forms:



(1) This type of error occurs particularly during the compilation or modification of a program, and the message is sent when the error is regarded as nothing to do with statement numbers.

(2) This error message is sent when an error is regarded as occuring in statement number XXXX specifically during program execution.

In addition, the following are types of error to be displayed.

SYNTAX

This occurs when there is an error in the program syntax.

MEMORY

When memory usable as a program is exceeded.

DATA

The range of data that can be handled by the computer is exceeded.

MISSMATCH

When variable for numerals and variable for strings are used in a mixed condition.

16FOR

When 16multi-loops or more are used for FOR-NEXT loops.

16GOSUB

When 16 or more levels of subroutine are used.

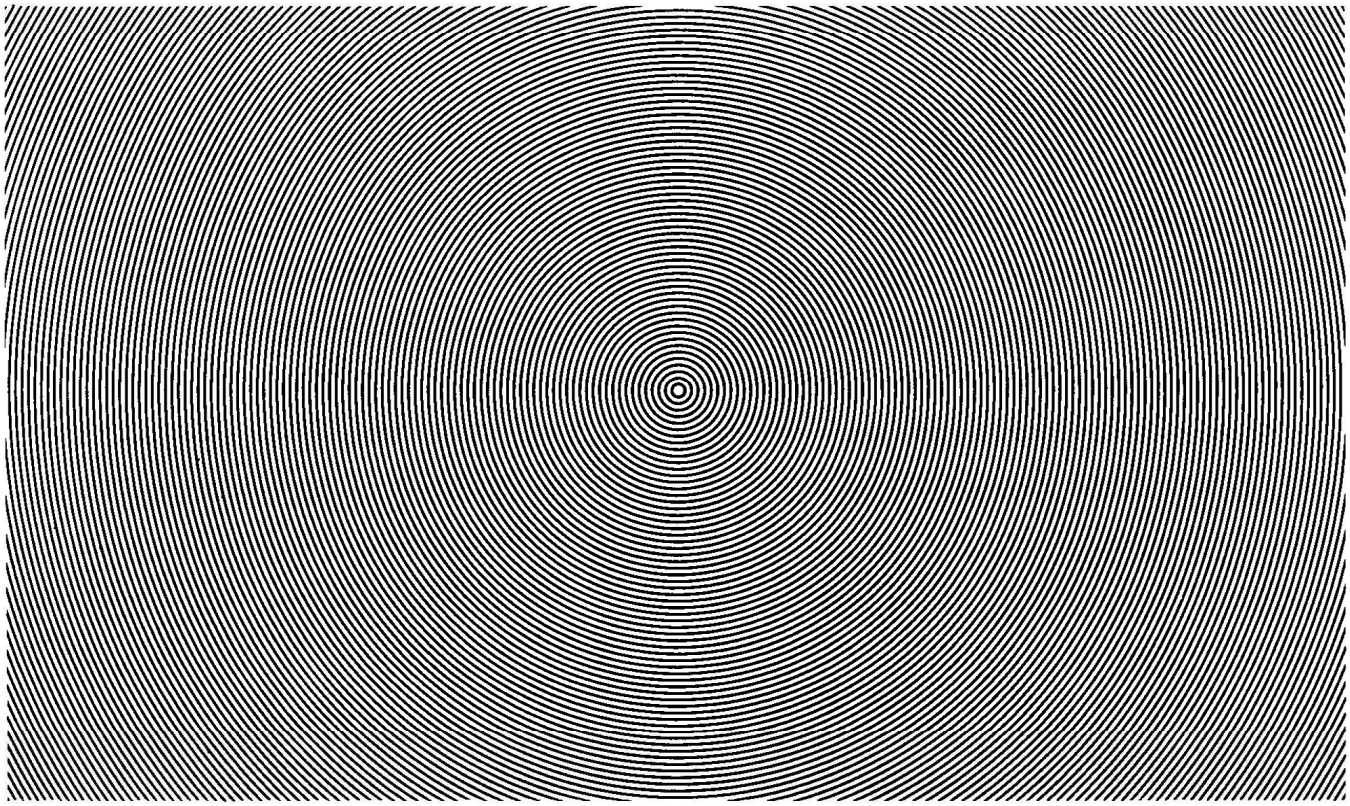
6FN

Another function definition can be used in the DEF FN function definition. However, when this multi-definition exceeds 6, an error occurs.

CONT

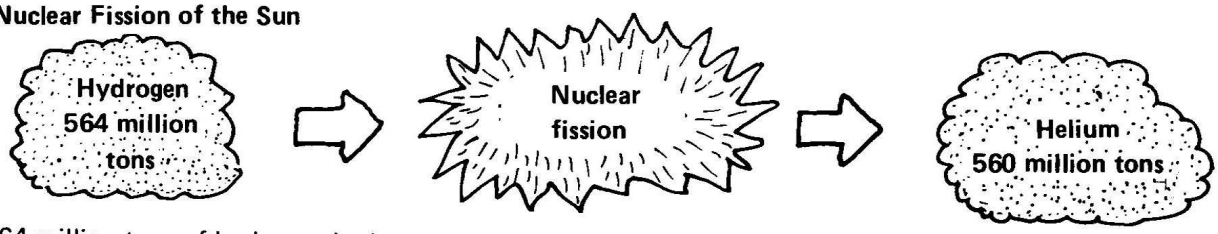
Even after STOP and END statements have been executed or a program execution is stopped by pressing the BREAK key, the program execution can be continued by the CONT command to re-start from the stop position. When the CONT command is used in any other cace, or the program is edited after it has been stopped, this error occurs. (Direct mode instructions, such as PRINT and LIST commands can be used in the above case.)

The Sun Becomes Lighter by 4 Million Tons per Second

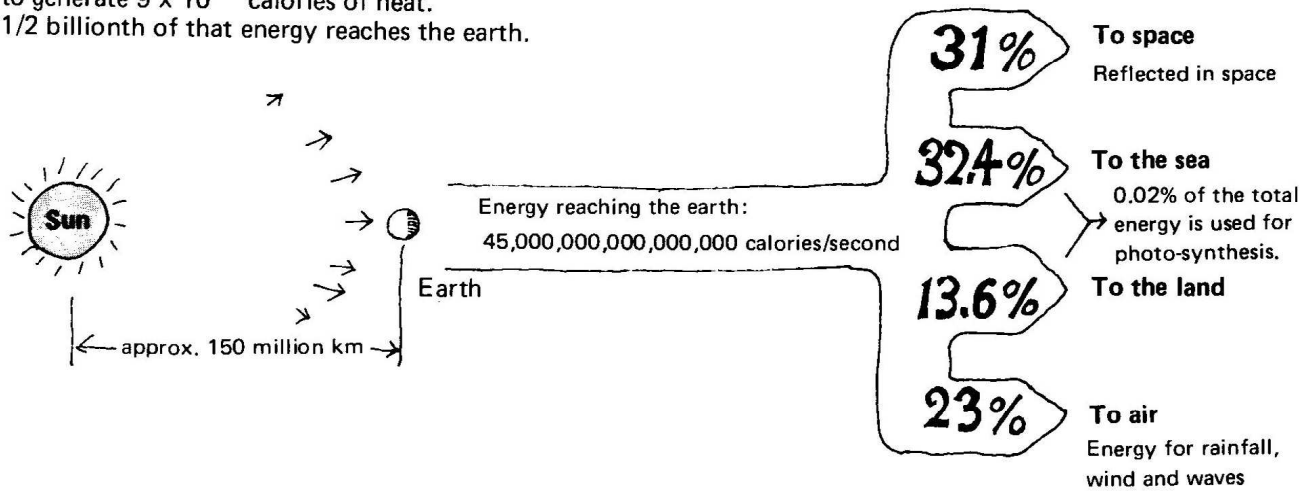


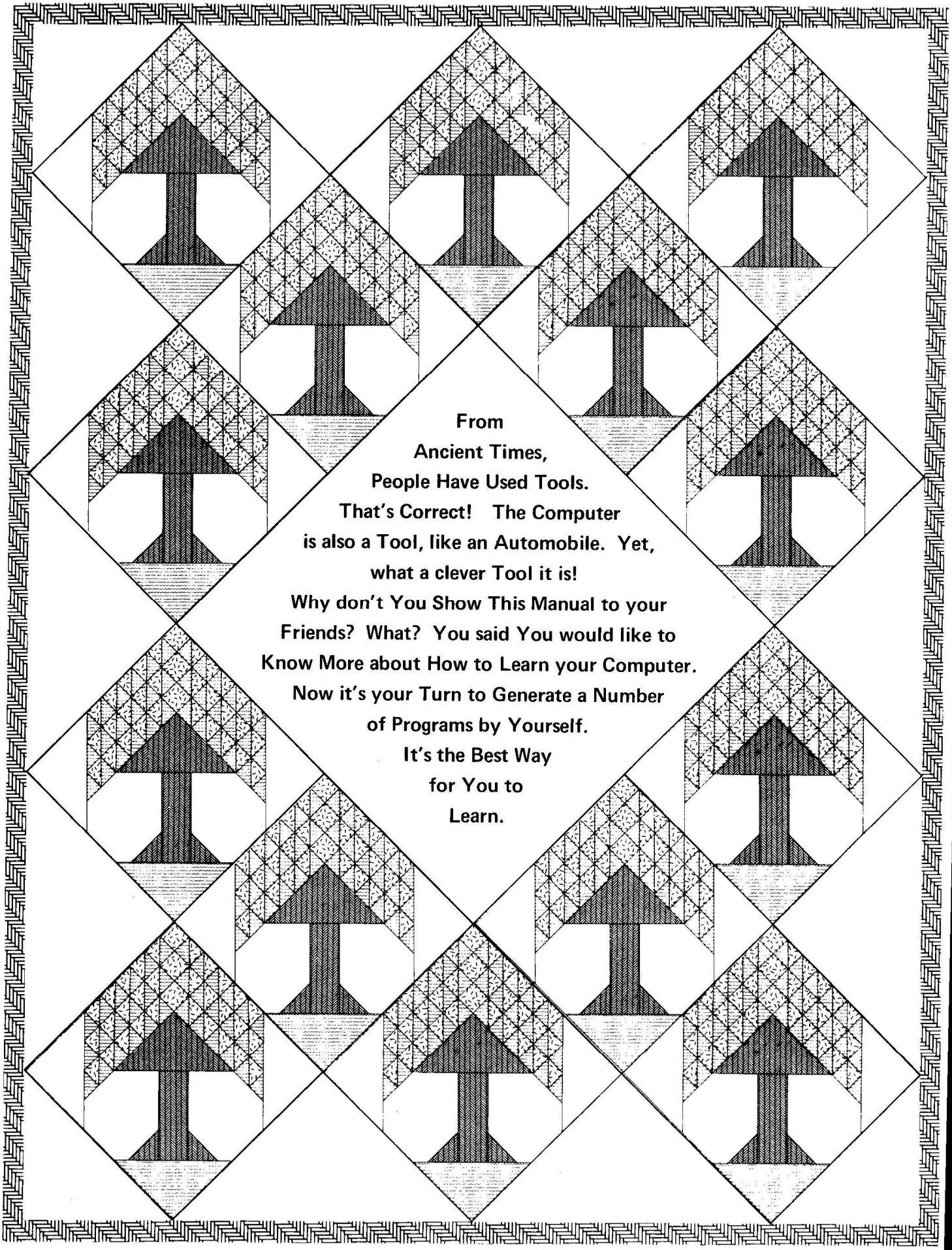
When substances are changed through nuclear reaction into energy, their mass decreases by that much. The sun changes hydrogen into helium through nuclear fission to generate 9×10^{25} calories of heat. The mass lost at that moment is 4 million tons. The sun grows us creatures while getting lighter. In fact, however, its energy radiating the earth, regarded as the only planet with creatures, is about 1/2 billionth of the sun's total. Yet, 1/3 of the energy radiation the earth is reflected into space. On earth, today, people are suffering from the shortage of energy for effective use. What a waste!

Nuclear Fission of the Sun



564 million tons of hydrogen is changed into 560 million tons of helium per second to generate 9×10^{25} calories of heat.
1/2 billionth of that energy reaches the earth.




From
Ancient Times,
People Have Used Tools.
That's Correct! The Computer
is also a Tool, like an Automobile. Yet,
what a clever Tool it is!
Why don't You Show This Manual to your
Friends? What? You said You would like to
Know More about How to Learn your Computer.
Now it's your Turn to Generate a Number
of Programs by Yourself.
It's the Best Way
for You to
Learn.

Display Code Table

The following is the display code of the MZ-80K. The code is based on the decimal system.

Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol
0	SP	32	0	64	SP	96	π	128	SP	160		192		224	
1	A	33	1	65		97	!	129	a	161		193		225	
2	B	34	2	66		98	"	130	b	162		194		226	
3	C	35	3	67		99	#	131	c	163		195		227	
4	D	36	4	68		100	$	132	d	164		196		228	
5	E	37	5	69		101	%	133	e	165		197		229	
6	F	38	6	70		102	&	134	f	166		198		230	
7	G	39	7	71		103	'	135	g	167		199		231	
8	H	40	8	72		104	(	136	h	168		200		232	
9	I	41	9	73		105	)	137	i	169		201		233	
10	J	42		74		106	+	138	j	170	β	202		234	
11	K	43		75		107	*	139	k	171	ü	203		235	
12	L	44	;	76		108		140	l	172	ö	204		236	
13	M	45	/	77		109		141	m	173	Ü	205		237	
14	N	46		78		110		142	n	174	Ä	206		238	
15	O	47	,	79		111		143	o	175	Ö	207		239	
16	P	48		80		112		144	p	176		208		240	SP
17	Q	49		81		113		145	q	177		209		241	
18	R	50		82		114		146	r	178		210		242	
19	S	51		83		115		147	s	179		211		243	
20	T	52		84		116		148	t	180		212		244	
21	U	53		85	@	117		149	u	181		213		245	
22	V	54		86		118		150	v	182		214		246	
23	W	55		87		119		151	w	183		215		247	
24	X	56		88		120		152	x	184		216		248	
25	Y	57		89		121		153	y	185		217		249	
26	Z	58		90		122		154	z	186		218		250	
27	£	59		91		123		155	ä	187		219	○	251	
28		60		92		124		156		188		220		252	
29		61		93		125		157		189		221		253	
30		62		94		126		158		190		222		254	
31		63		95		127		159		191		223		255	

Note: SP represents a space or blank.

Special Character Code and Memory Map

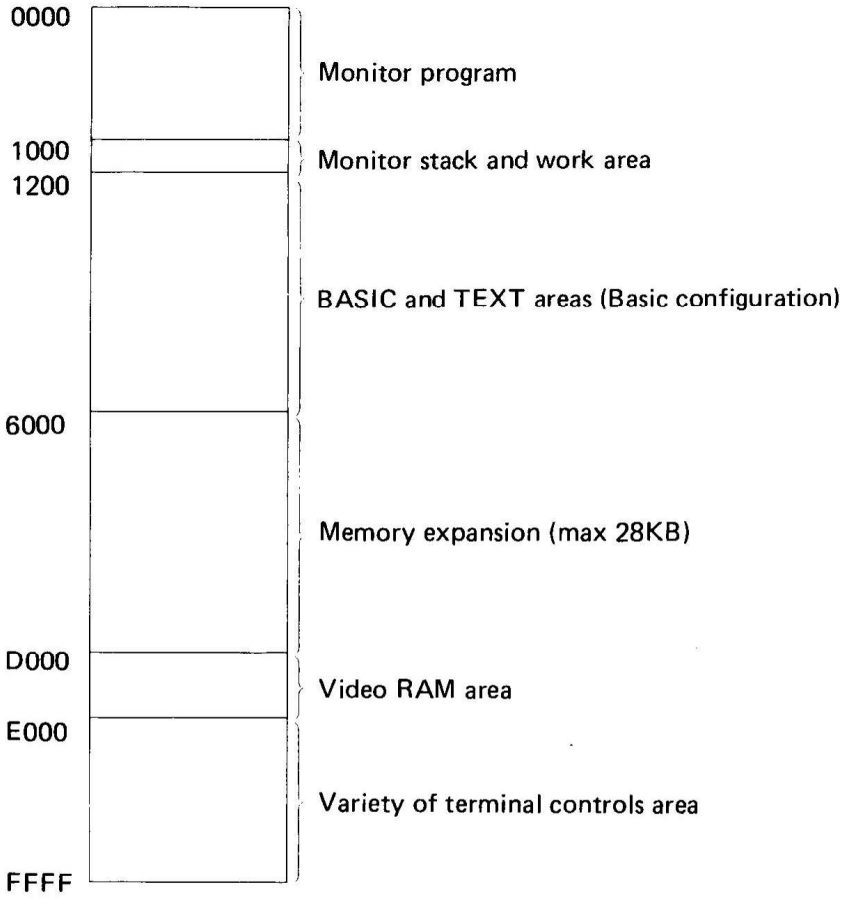
Special Character Code

The following special codes function to shift the cursor, set home and clear the TV screen when the PRINT statement with quotation marks " " has been executed. The MZ-80K has 6 such codes based on the decimal system.

Code	Symbol	Function	Key
193	↓	Cursor shifts down by 1 character space	↓
194	↑	Cursor shifts up by 1 character space	↑
195	→	Cursor shifts to right by 1 character space	→
196	←	Cursor shifts to left by 1 character space	←
197	H	Cursor shifts to top left corner on TV screen. (home)	HOME
198	C	Clears TV screen and home	CLR

Memory Map

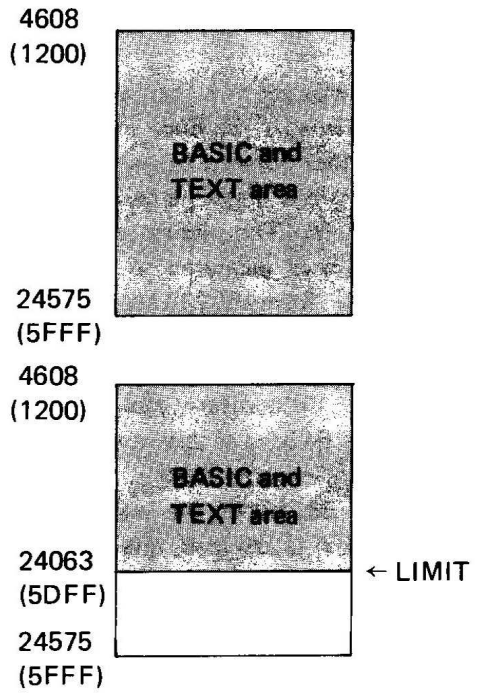
When BASIC is installed with RAM standard specifications, the area partitions are as outlined below:



Linkage to Machine Language

The use of POKE command makes possible direct writing of the machine language into memory address, while the USR statement can shift program control to a specified memory address. It is, however, dangerous to use the POKE command carelessly. Look at the diagram at right. RAM area is assumed to range from 4608 to 24575 in decimal addresses. The parenthesis ( ) in the diagram indicates a hexadecimal address. Usually, decimal address from 4608 to 24575 is the BASIC and TEXT area (Top diagram). Therefore, when the POKE command is executed under this condition, the contents of the already stored program are destroyed. Because of this, using the command caselessly is dangerous.

When using the POKE command, therefore, it is wise to ensure a safety memory address region. At this time, the LIMIT statement can be used. It sets the maximal memory address used in the BASIC instructions, and executes the following statement, for example.




```
LIMIT      24063
```

This limits the maximal memory address used in the BASIC instructions to 24063. At this point, the BASIC and TEXT area ranges decimal addresses from 4608 to 24063, and decimal addresses from 24064 to 24575 are ensured as a safety region. Therefore, this area can be an assigned address by the POKE command (bottom diagram). This does not destroy other programs and is a very reliable method. A simple program is shown as follows:

```
10 PRINT "■"
20 LIMIT 24063
30 GOSUB 100
40 USR (24064)
50 END
100 FOR A = 24064 TO 24087
110 READ D : POKE A, D : NEXT A
120 RETURN
130 DATA 197, 213, 229, 22, 0, 33, 0, 208, 1, 232
140 DATA 3, 114, 35, 20, 11, 120, 177, 194, 11
150 DATA 94, 225, 209, 193, 201
```

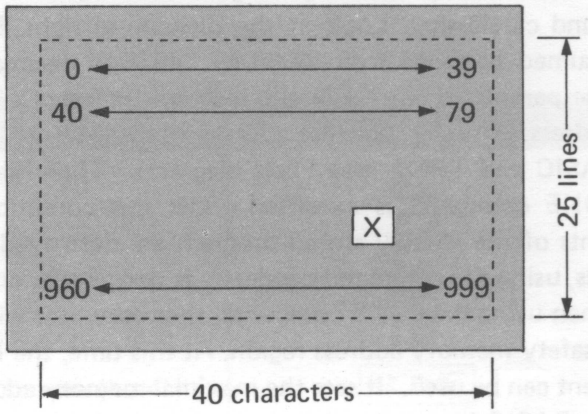
This is the program for display of characters on the TV screen. Using the machine language program, the display code is stored in the video RAM memory. The machine language as data is written into memory by the POKE command. Accordingly, first use the LIMIT statement, in order to limit the maximal memory (decimal address) used in the BASIC to 24063. Using the POKE command, the machine language is stored into memory starting at decimal address 24064.

With execution of LIMIT MAX statement, BASIC and TEXT area is put pack.

TV Screen Constitution and Special Control Command

TV Screen Constitution

As shown in the diagram at right, the TV screen consists of picture elements, 25 lines and 40 characters. Each element further consists of a 8 x 8 dot matrix. Such dot combinations display a certain character on each picture element. The picture element positions on the TV screen is allocated as video RAM to decimal addresses from 53248 to 57343 in the internal memory. To display a character at any desired position on the TV screen, keying-in is made using the cursor. In addition, however, the POKE command can also be used as described below:
For example, displacement X ($0 \leqq X \leqq 999$) from memory address 53248 of the picture element at any desired position is assigned, and the display code for the desired character is assumed Y ($0 \leqq Y \leqq 255$).



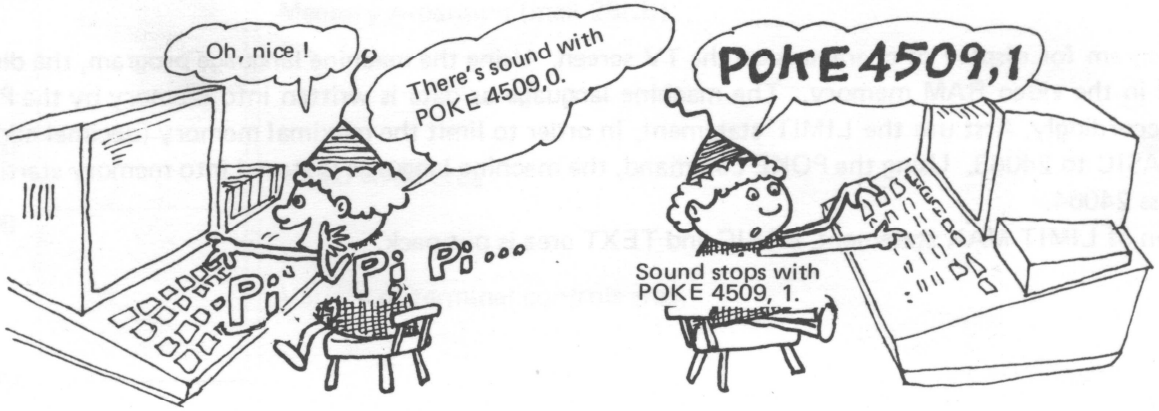
POKE 53248 + X, Y

With the above command, the character is displayed at X position on the TV screen.

Special Control Commands

Including the above POKE command, the MZ-80K includes the following special control commands.

POKE 53248 + X, Y	: Displays a character corresponding to display code Y at picture element X position.
POKE 4509, 0	: Entry bell sounds when a key is pressed.
POKE 4509, 1	: Stops bell sound when a key is pressed.
USR (62)	: The bell sounds when this statement is executed.



ASCII Code Table

The following are the ASCII codes for characters:

Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol
32	SP	64	@	96		128	SP	160	q	192	SP	224	
33	!	65	A	97		129		161	a	193		225	
34	"	66	B	98		130		162	z	194		226	
35	#	67	C	99		131		163	w	195		227	
36	$	68	D	100		132		164	s	196		228	
37	%	69	E	101		133		165	u	197		229	
38	&	70	F	102		134		166	i	198		230	
39	'	71	G	103		135		167		199		231	
40	(	72	H	104		136		168	Ö	200		232	
41	)	73	I	105		137		169	k	201		233	
42	*	74	J	106		138		170	f	202		234	
43	+	75	K	107		139		171	v	203		235	
44	,	76	L	108		140		172		204		236	
45	-	77	M	109		141		173	ü	205		237	
46	.	78	N	110		142		174	ß	206		238	
47	/	79	O	111		143		175	j	207		239	
48	0	80	P	112		144		176	n	208		240	
49	1	81	Q	113		145		177		209		241	
50	2	82	R	114		146	e	178	Ü	210		242	
51	3	83	S	115		147		179	m	211		243	
52	4	84	T	116		148		180		212		244	
53	5	85	U	117		149		181		213		245	
54	6	86	V	118		150	t	182		214		246	
55	7	87	W	119		151	g	183	o	215		247	
56	8	88	X	120		152	h	184	l	216		248	
57	9	89	Y	121		153		185	Ä	217		249	
58	:	90	Z	122		154	b	186	ö	218		250	
59	;	91	[	123		155	x	187	ä	219		251	
60	<	92	\	124		156	d	188		220		252	
61	=	93	]	125		157	r	189	y	221		253	
62	>	94		126		158	p	190	¥	222		254	
63	?	95		127		159	c	191		223		255	

Note: The code is based on the decimal system. SP represents a space.

Z-80 Instruction List

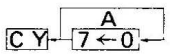
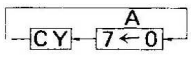
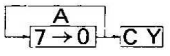
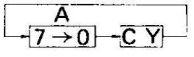
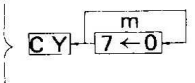
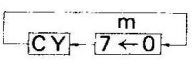
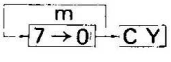
The MZ-80K uses the Z-80 as the CPU, and its instruction list is given below for information.

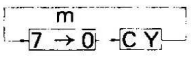
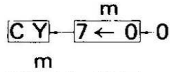
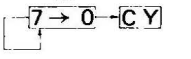
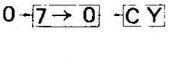
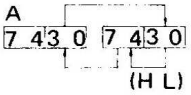
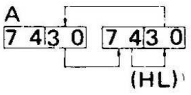
Mnemonic	Symbolic Operation	Instruction Code 76 543 210
LD r, r'	r ← r'	01 r r'
LD r, n	r ← n	00 r 110 ← n →
LD r, (HL)	r ← (HL)	01 r 110
LD r, (IX + d)	r ← (IX + d)	11 011 101 01 r 110 ← d →
LD r, (IY + d)	r ← (IY + d)	11 111 101 01 r 110 ← d →
LD (HL), r	(HL) ← r	01 110 r
LD (IX + d), r	(IX + d) ← r	11 011 101 01 110 r ← d →
LD (IY + d), r	(IY + d) ← r	11 111 101 01 110 r ← d →
LD (HL), n	(HL) ← n	00 110 110 ← n →
LD (IX + d), n	(IX + d) ← n	11 011 101 00 110 110 ← d → ← n →
LD (IY + d), n	(IY + d) ← n	11 111 101 00 110 110 ← d → ← n →
LD A, (BC)	A ← (BC)	00 001 010
LD A, (DE)	A ← (DE)	00 011 010
LD A, (nn)	A ← (nn)	00 111 010 ← n → ← n →
LD (BC), A	(BC) ← A	00 000 010
LD (DE), A	(DE) ← A	00 010 010
LD (nn), A	(nn) ← A	00 110 010 ← n → ← n →
LD A, I	A ← I	11 101 101 01 010 111
LD A, R	A ← R	11 101 101 01 011 111
LD I, A	I ← A	11 101 101 01 000 111
LD R, A	R ← A	11 101 101 01 001 111

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
LD dd, nn	dd ← nn	00 dd0 011 ← n → ← n →
LD IX, nn	IX ← nn	11 011 101 00 100 001 ← n → ← n →
LD IY, nn	IY ← nn	11 111 101 00 100 001 ← n → ← n →
LD HL, (nn)	H ← (nn + 1) L ← (nn)	00 101 010 ← n → ← n →
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	11 101 101 01 dd1 011 ← n → ← n →
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	11 011 101 00 101 010 ← n → ← n →
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	11 111 101 00 101 010 ← n → ← n →
LD (nn), HL	(nn + 1) ← H (nn) ← L	00 100 010 ← n → ← n →
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	11 101 101 01 dd0 011 ← n → ← n →
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	11 011 101 00 100 010 ← n → ← n →
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	11 111 101 00 100 010 ← n → ← n →
LD SP, HL	SP ← HL	11 111 001
LD SP, IX	SP ← IX	11 011 101 11 111 001
LD SP, IY	SP ← IY	11 111 101 11 111 001
PUSH qq	(SP−2) ← qq_L (SP−1) ← qq_H	11 qq0 101

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
PUSH IX	$(SP-2) \leftarrow IX_L$	11 011 101
	$(SP-1) \leftarrow IX_H$	11 100 101
PUSH IY	$(SP-2) \leftarrow IY_L$	11 111 101
	$(SP-1) \leftarrow IY_H$	11 100 101
POP qq	$qq_H \leftarrow (SP+1)$	11 qq0 001
	$qq_L \leftarrow (SP)$	
POP IX	$IX_H \leftarrow (SP+1)$	11 011 101
	$IX_L \leftarrow (SP)$	11 100 001
POP IY	$IY_H \leftarrow (SP+1)$	11 111 101
	$IY_L \leftarrow (SP)$	11 100 001
Exchange group, block transfer and search group		
EX DE, HL	$DE \leftrightarrow HL$	11 101 011
EX AF, AF'	$AF \leftrightarrow AF'$	00 001 000
EXX	$(BC) \leftrightarrow (BC')$	11 011 001
	$(DE) \leftrightarrow (DE')$	
	$(HL) \leftrightarrow (HL')$	
EX (SP), HL	$H \leftrightarrow (SP+1)$	11 100 011
	$L \leftrightarrow (SP)$	
EX (SP), IX	$IX_H \leftrightarrow (SP+1)$	11 011 101
	$IX_L \leftrightarrow (SP)$	11 100 011
EX (SP), IY	$IY_H \leftrightarrow (SP+1)$	11 111 101
	$IY_L \leftrightarrow (SP)$	11 100 011
LDI	$(DE) \leftarrow (HL)$	11 101 101
	$DE \leftarrow DE+1$	10 100 000
	$HL \leftarrow HL+1$	
	$BC \leftarrow BC-1$	
LDIR	$(DE) \leftarrow (HL)$	11 101 101
	$DE \leftarrow DE+1$	10 110 000
	$HL \leftarrow HL+1$	
	$BC \leftarrow BC-1$	
	Repeat until BC=0	
LDD	$(DE) \leftarrow (HL)$	11 101 101
	$DE \leftarrow DE-1$	10 101 000
	$HL \leftarrow HL-1$	
	$BC \leftarrow BC-1$	
LDDR	$(DE) \leftarrow (HL)$	11 101 101
	$DE \leftarrow DE-1$	10 111 000
	$HL \leftarrow HL-1$	
	$BC \leftarrow BC-1$	
	Repeat until BC=0	
CPI	$A-(HL)$	11 101 101
	$HL \leftarrow HL+1$	10 100 001
	$BC \leftarrow BC-1$	
CPIR	$A-(HL)$	11 101 101
	$HL \leftarrow HL+1$	10 110 001
	$BC \leftarrow BC-1$	
	Repeat until BC = 0 or A = (HL)	
CPD	$A-(HL)$	11 101 101
	$HL \leftarrow HL-1$	10 101 001
	$BC \leftarrow BC-1$	

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
CPDR	A - (HL)	11 101 101
	$HL \leftarrow HL-1$	10 111 001
	$BC \leftarrow BC-1$	
	Repeat until BC = 0 or A = (HL)	
8-bit arithmetic and logic group		
ADD A, r	$A \leftarrow A+r$	10 [000] r
ADD A, n	$A \leftarrow A+n$	11 [000] 110
		← n →
ADD A, (HL)	$A \leftarrow A+(HL)$	10 [000] 110
ADD A, (IX + d)	$A \leftarrow A+(IX+d)$	11 011 101
		10 [000] 110
		← d →
ADD A, (IY + d)	$A \leftarrow A+(IY+d)$	11 111 101
		10 [000] 110
		← d →
ADC A, s	$A \leftarrow A+s+CY$	[001]
SUB s	$A \leftarrow A-s$	[010]
SBC A, s	$A \leftarrow A-s-CY$	[011]
AND s	$A \leftarrow A \wedge s$	[100]
OR s	$A \leftarrow A \vee s$	[110]
XOR s	$A \leftarrow A \oplus s$	[101]
CP s	$A-s$	[111]
INC r	$r \leftarrow r+1$	00 r [100]
INC (HL)	$(HL) \leftarrow (HL)+1$	00 110 [100]
INC (IX + d)	$(IX+d)$	11 011 101
	$\leftarrow (IX+d)+1$	00 110 [100]
		← d →
INC (IY + d)	$(IY+d)$	11 111 101
	$\leftarrow (IY+d)+1$	00 110 [100]
		← d →
DEC m	$m \leftarrow m-1$	[101]
General Purpose Arithmetic and CPU Control Group		
DAA	Correct decimal contents of A after addition and subtraction	00 100 111
CPL	$A \leftarrow \overline{A}$	00 101 111
NEG	$A \leftarrow \overline{A}+1$	11 101 101
		01 000 100
CCF	$CY \leftarrow \overline{CY}$	00 111 111
SCF	$CY \leftarrow 1$	00 110 111
NOP	Nothing is executed, but PC contents incremented.	00 000 000
HALT	CPU halted	01 110 110
DI	$IFF \leftarrow 0$	11 110 011
EI	$IFF \leftarrow 1$	11 111 011
IM0	Set interrupt mode 0.	11 101 101
		01 000 110

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
IM1	Set interrupt mode 1	11 101 101 01 010 110
IM2	Set interrupt mode 2	11 101 101 01 011 110
ADD HL, ss	HL ← HL + ss	00 ss1 001
ADC HL, ss	HL ← HL + ss + CY	11 101 101 00 ss1 010
SBC HL, ss	HL ← HL − ss − CY	11 101 101 01 ss0 010
ADD IX, pp	IX ← IX + pp	11 011 101 00 pp1 001
ADD IY, rr	IY ← IY + rr	11 111 101 00 rr1 001
INC ss	ss ← ss + 1	00 ss0 011
INC IX	IX ← IX + 1	11 011 101 00 100 011
INC IY	IY ← IY + 1	11 111 101 00 100 011
DEC ss	ss ← ss − 1	00 ss1 011
DEC IX	IX ← IX − 1	11 011 101 00 101 011
DEC IY	IY ← IY − 1	11 111 101 00 101 011
RLCA		00 000 111
RLA		00 010 111
RRCA		00 001 111
RRA		00 011 111
RLC r		11 001 011 00 000 r
RLC (HL)		11 001 011 00 000 110
RLC (IX + d)		11 011 101 11 001 011 ← d → 00 000 110
RLC (IY + d)		11 111 101 11 001 011 ← d → 00 000 110
RL m		010
RRC m		001

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
RR m		011
SLA m		100
SRA m		101
SRL m		111
RLD		11 101 101 01 101 111
RRD		11 101 101 01 100 111
BIT b, r	Z ← $\overline{r_b}$	11 001 011 01 b r
BIT b, (HL)	Z ← $\overline{(HL)_b}$	11 001 011 01 b 110
BIT b, (IX + d)	Z ← $\overline{(IX + d)_b}$	11 011 101 11 001 011 ← d → 01 b 110
BIT b, (IY + d)	Z ← $\overline{(IY + d)_b}$	11 111 101 11 001 011 ← d → 01 b 110
SET b, r	r_b ← 1	11 001 011 11 b r
SET b, (HL)	$(HL)_b$ ← 1	11 001 011 11 b 110
SET b, (IX + d)	$(IX + d)_b$ ← 1	11 011 101 11 001 011 ← d → 11 b 110
SET b, (IY + d)	$(IY + d)_b$ ← 1	11 111 101 11 001 011 ← d → 11 b 110
RES b, m	m_b ← 0	10
JP nn	PC ← nn	11 000 011 ← n → ← n →
JP cc, nn	If condition cc is true, PC ← nn	11 cc 010 ← n → ← n →
JR e	PC ← PC + e	00 011 000 ← e - 2 →
JR C, e	If C = 0, Continue If C = 1, PC ← PC + e	00 111 000 ← e - 2 →

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
JR Z, e	If Z = 0,	00 101 000
	Continue	← e - 2 →
	If Z = 1,	
	PC ← PC + e	
JR NC, e	If C = 1,	00 110 000
	Continue	← e - 2 →
	If C = 0,	
	PC ← PC + e	
JR NZ, e	If Z = 1,	00 100 000
	Continue	← e - 2 →
	If Z = 0,	
	PC ← PC + e	
JP (HL)	PC ← HL	11 101 001
JP (IX)	PC ← IX	11 011 101
		11 101 001
JP (IY)	PC ← IY	11 111 101
		11 101 001
DJNZ e	B ← B − 1	00 010 000
	If B ≠ 0,	← e - 2 →
	Continue	
	If B ≠ 0,	
	PC ← PC + e	
Call and return group		
CALL nn	$(SP - 1) \leftarrow PC_H$	11 001 101
	$(SP - 2) \leftarrow PC_L$	← n →
	PC ← nn	← n →
CALL cc, nn	If condition cc is true,	11 cc 100
	same as CALL nn.	← n →
	If condition cc is false, continues.	← n →
RET	$PC_L \leftarrow (SP)$	11 001 001
	$PC_H \leftarrow (SP + 1)$	
RET cc	If condition cc is true,	11 cc 000
	same as RET	
	If cc is false, continues	
RETI	Returns from interrupt	11 101 101
		01 001 101
RETN	Return from $\overline{NMI}$	11 101 101
		01 000 101
RST p	$(SP - 1) \leftarrow PC_H$	11 t 111
	$(SP - 2) \leftarrow PC_L$	
	$PC_H \leftarrow 0$	
	$PC_L \leftarrow p$	

Mnemonic	Symbolic Operation	Instruction Code 76 543 210
Input / Output group		
IN A, (n)	A ← (n)	11 011 011
		← n →
IN r, (C)	r ← (C)	11 101 101
		01 r 000
INI	(HL) ← (C)	11 101 101
	B ← B − 1	10 100 010
	HL ← HL + 1	
INIR	(HL) ← (C)	11 101 101
	B ← B − 1	10 110 010
	HL ← HL + 1	
	Repeat until B=0	
IND	(HL) ← (C)	11 101 101
	B ← B − 1	10 101 010
	HL ← HL − 1	
INDR	(HL) ← (C)	11 101 101
	B ← B − 1	10 111 010
	HL ← HL − 1	
	Repeat until B=0	
OUT (n), A	(n) ← A	11 010 011
		← n →
OUT (C), r	(C) ← r	11 101 101
		01 r 001
OUTI	(C) ← (HL)	11 101 101
	B ← B − 1	10 100 011
	HL ← HL + 1	
OTIR	(C) ← (HL)	11 101 101
	B ← B − 1	10 110 011
	HL ← HL + 1	
	Repeat until B=0	
OUTD	(C) ← (HL)	11 101 101
	B ← B − 1	10 101 011
	HL ← HL − 1	
OTDR	(C) ← (HL)	11 101 101
	B ← B − 1	10 111 011
	HL ← HL − 1	
	Repeat until B=0	

Note: The following shows the designation of the symbols used in the instruction list:

r, r'	Register
000	B
001	C
010	D
011	E
100	H
101	L
111	A

dd, ss	Register Pair
00	BC
01	DE
10	HL
11	SP

qq	Register Pair
00	BC
01	DE
10	HL
11	AF

pp	Register Pair
00	BC
01	DE
10	IX
11	SP

rr	Register Pair
00	BC
01	DE
10	IY
11	SP

∧: Logical AND operation
∨: Logical OR operation
⊕: Logical exclusive-OR operation

b	Bit set
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

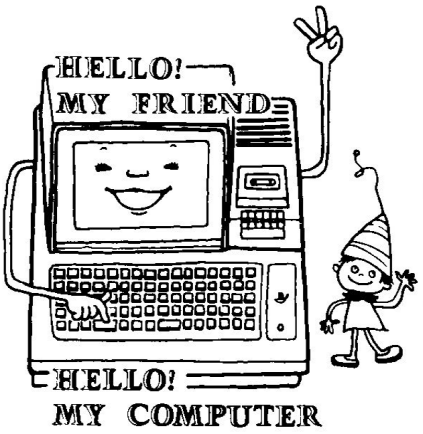
cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

s : r, n, (HL), (IX + d), (IY + d)
CY: carry flip-flop
$(\text{Register Pair})_H$: upper 8 bits of register

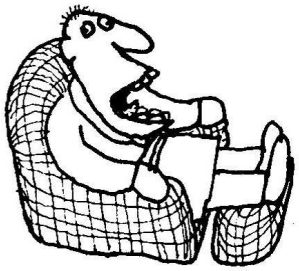
m : r, (HL), (IX + d), (IY + d)
m_b : bit b (0 to 7) or location m
$(\text{Register Pair})_L$: Lower 8 bits of register

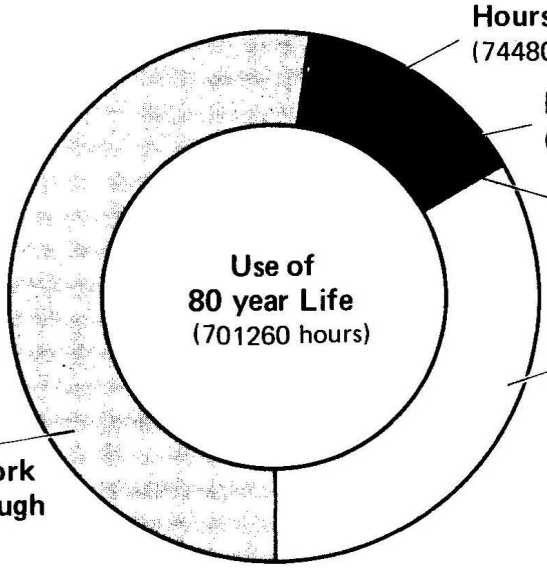
Instruction code of ADC, SUB, SBC, AND, OR, XOR, CP to the mnemonic code is the contents ☐ replaced by those of ADD group ☐ .
Instruction code of DEC is the content replaced by those of INC group ☐
Similar processing is made for instruction code ☐ in rotate, shift group and bit set, reset and test group.

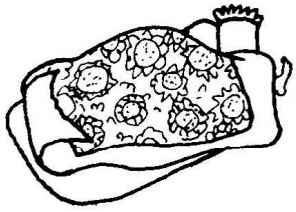

HELLO!
MY FRIEND
HELLO!
MY COMPUTER

701,260 Hours

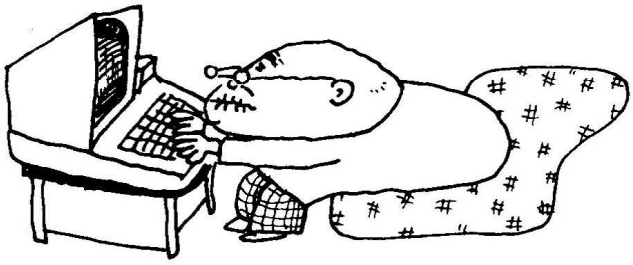
You have mastered your computer to use it as if it were part of your brains, haven't you ? What did you say ? You are too busy to have time for that. Indeed, we are living in this busy world, aren't we ?
By the way, let's predict, using the computer, what a busy life you are leading. Calculations are made for a life of 80 years that is a little longer than the English average. For education, 16 years are spent at infant school, prinary school, high school and university or college. 245 days a year are for going to school and 5 hours a day for lessons. After school, 1.5 hours are for home work every day throughout the year. After graduation from university, 8 hours a day for 245 days a year are for work as a salaried man with 8 hours a day for sleep. 365.24 days a year are assumed. Based on the above, calculations are made, with results as follows:

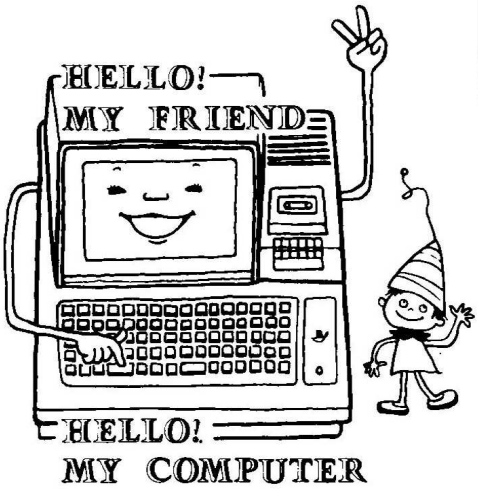






How did you enjoy the calculations ? During a life of 80 years, studying accounts for 4.1% and working 10.6%. Why not analyze and predict the use of your own time for you future reference and consideration ?
Don't forget to add your time working on your computer.




HELLO!
MY FRIEND
HELLO!
MY COMPUTER

Precautions in Operation

- **Power Cord**

  Don't pin the power cord under a desk or chair, nip or damage it. It is dangerous to use the power cord with damage of any kind. Hold the plug whenever the power cord is disconnected.

- **Power Voltage**

  The computer operates on the local voltage. Unstable power voltages will cause the unit to malfunction, resulting in incapability to give high performance, for which it is designed. For any problem of malfunction, consult the dealer from where the unit was purchased.

- **Ventilation**

  To prevent any temperature rise, the cabinets are provided with air vents. Do not keep the unit in a poorly ventilated place, cover it with cloth, place it on a carpet or soft surface that could impede ventilation.

- **Humidity and Dust**

  Make sure that the unit is free from humidity or dust, particularly bathrooms or kitchen. Such places could cause the unit to malfunction.

- **High Temperature**

  Place the unit on any convenient location that should be out of direct sunlight and away from heat sources such as airconditioners, heaters. Such sources may cause the cabinets and internal component parts to be damaged.

- **Moisture and Impurities**

  It is dangerous to use the unit with moisture, liquid or metals, such as needles and pins included inside. Make certain that the unit is free from impurities. If moisture or liquid enters, unplug the power cord immediately, and contact the dealer from where the unit was purchased.

- **Impact or Shock**

  The unit is made up of precision electronic parts. Do not drop, hit or give any impact to the unit.

- **While not in Use for Long Periods**

  Be sure to disconnect the power cord from the power outlet while not in use for long periods.

- **Cleaning**

  Clean the unit with a soft cloth damped with water or cleanser. Do not clean the unit with volatile liquid, such as benzine and thinner, or apply an insecticide. This could cause the case to discolour or the unit to malfunction.

- **Location**

  Do not locate the unit under extremely high or low temperature or under great charges in humidity. The unit should also be away from vibration and dust. Nothing should be on top of it.

- **Operation and Maintenance**

  Do not use or store the unit with the cabinet opened or with the top cover removed. This could cause an electric shock or malfunction of the unit.

- **Noise Wave**

  If the unit is used near a radio or TV, its operation may adversely interfere with the radio or TV. In addition, the unit may receive influences from sources generating strong magnetic forces like a TV. Therefore, be sure to use the unit 2 to 3 meters away from such sources.

- **Power ON/OFF**

  ON/OFF operation of the power switch should be conducted at an interval of 10 seconds or more. This is to ensure operation of the built-in microcomputer.

- **Cassette Handling**

  If stained, the recording and playback heads inside the cassette tape recorder make it impossible to record and reproduce correct data.

  It is recommended that the unit be cleaned once a month. Cleaning tape available on the market will be convenient. When the tape is not run, be sure to press the stop button.