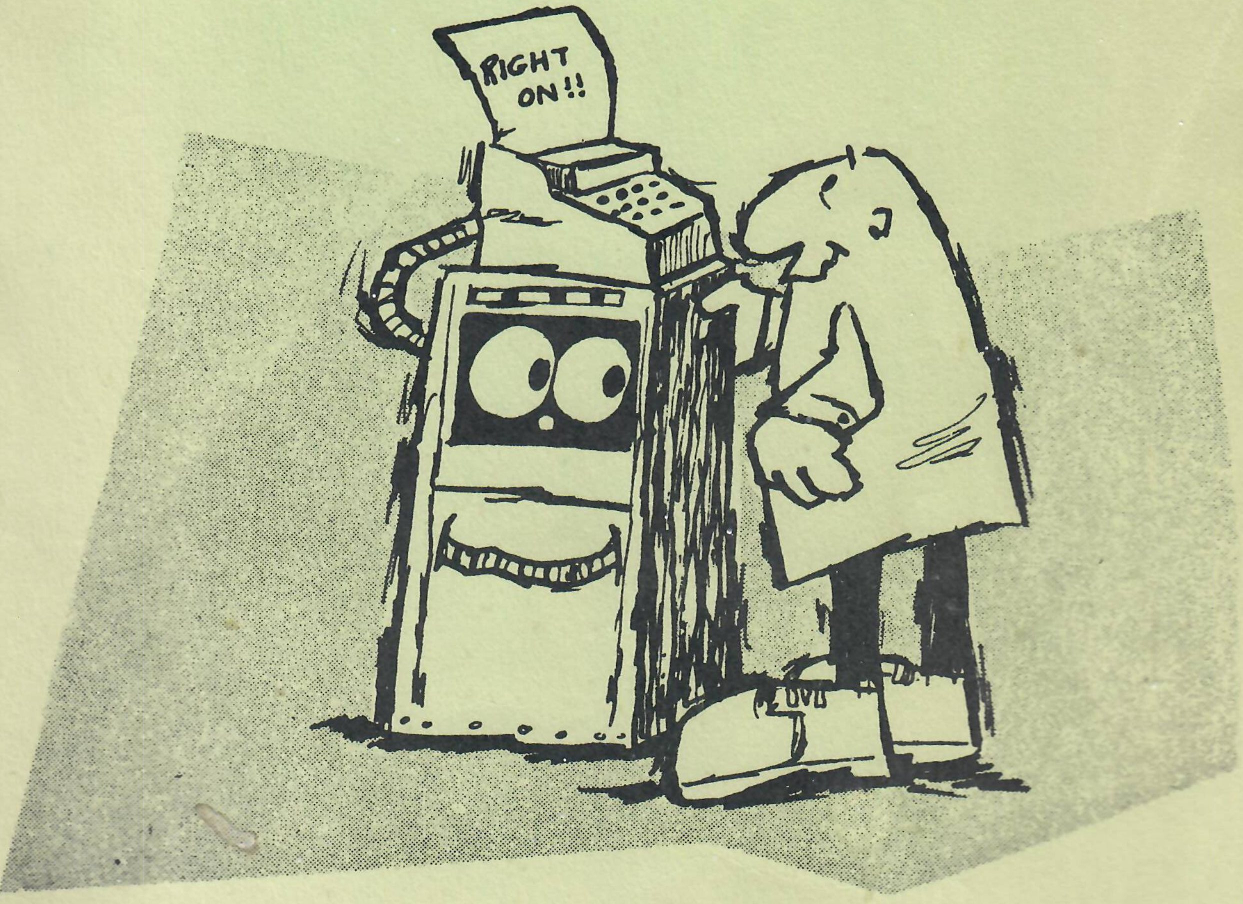
$2.00

MY COMPUTER LIKES ME *

by BOB ALBRECHT



*when i speak in BASIC

CONTENTS

TTY

This book is about people, computers and a programming language called BASIC. We will communicate with a computer, in the BASIC language, about population problems.

We will use a *teletypewriter.*

Teletypewriters are the Volkswagens of computer terminals . . . rugged, dependable, inexpensive, ugly and noisy!

We assume:

you know how to operate a teletypewriter, or

someone will show you how to use a teletypewriter, or

you can figure out how to use a teletypewriter (it's easy).

REMEMBER THIS: Many people use "TTY" as an abbreviation for "teletypewriter." We will too.

We can't answer all your questions in this book, but you, the TTY and the computer can answer most of them.

EXPERIMENT! GAMBLE! GUESS,...
THEN TRY IT!

Begin

We use the TTY (teletypewriter . . . remember?) to send messages to the computer. In turn, the computer sends messages to the TTY for us to read.

Begin! Type your name on the TTY and press the **RETURN** key. The computer will probably type

```
WHAT?
```

THE MESSAGE VARIES FROM COMPUTER TO COMPUTER. BUT THE POINT IS, THE COMPUTER DID NOT UNDERSTAND YOU.

or perhaps

```
???
```

or maybe

```
SYNTAX ERR0R
```

(Look up SYNTAX in your dictionary.)

So try this . . . you press the keys . . . you do the typing and remember, finish each line by pressing the **RETURN** key.

You type

```
SCR

10 PRINT "MY HUMAN UNDERSTANDS ME"
20 END
```

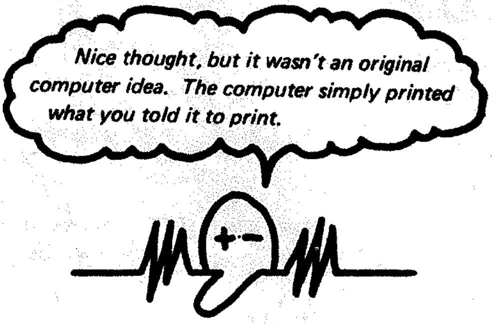
"Don't forget the quotation marks."

then type

```
RUN
```

and the computer will type

```
MY HUMAN UNDERSTANDS ME
```



Let's review what happened. First, you typed

```
SCR
```

and pressed the **RETURN** key. This told the computer to SCRatch any old information in its memory. It's sort of like erasing the blackboard before you start writing on it.

Next, you typed a short *program*, consisting of two *statements.*

```
10 PRINT "MY HUMAN UNDERSTANDS ME"
20 END
```

As you typed the program, it was stored into the computer's memory. After entering the program, you then typed

```
RUN
```

The computer ran the program and typed

```
MY HUMAN UNDERSTANDS ME
```

Then it stopped.

> The statement: `10 PRINT "MY HUMAN UNDERSTANDS ME"`
>
> tells the computer to print this message. The message is enclosed in quotation marks.
>
> The statement: `20 END`
>
> tells the computer to stop.

A *statement* begins with a line number. A line number is a *whole number* between 1 and 9999, inclusive.

```
10 PRINT "MY HUMAN UNDERSTANDS ME"
```

line number

When you type RUN and press **RETURN**, the computer obeys each statement in the program, in line number order.

Still your turn. Try this one . . . type

```
SCR                            SCRatch the preceding program.

10 PRINT "7 + 5"               Enter the new program.
20 END

RUN                            RUN the new program.

7 + 5              <--------   The computer types what you tell it to type.
```

Next . . . let's *replace* Line 10 with a new Line 10. (Retype the line, including the line number.)

```
10 PRINT 7 + 5     <--  No quotation marks.
```

Then tell the computer to LIST the current program.

```
LIST

10 PRINT 7 + 5   <-------- Here is the new Line 10,
20 END           <-------- and the old Line 20.

RUN                        RUN it.

 12                        This time the computer does the arithmetic.
```

The statement

PRINT 7 + 5 **without “ ”**

tells the computer to evaluate the *numerical expression* 7 + 5 (that is, do the arithmetic) and print the result as a decimal numeral.

The statement

PRINT "7 + 5" **with “ ”**

tells the computer to print the *string* enclosed in quotation marks *exactly as it appears.* No arithmetic is performed.

Strings? Numerical expressions?

```
PRINT "MY HUMAN UNDERSTANDS ME"
```

This is a string. It is enclosed in quotation marks.

```
PRINT "7 + 5"          PRINT 7 + 5
```

This is a string. It is enclosed in quotation marks.

This is not a string. It is a numerical expression.

Your turn again. Try these.

```
SCR

10 PRINT "7 + 5=" , 7 + 5
20 END
RUN

7 + 5=          12
```

Note the comma.

Now, replace Line 10 like this

```
10 PRINT "7 + 5=" ; 7 + 5
```

Note the semicolon.

Then LIST the modified program and RUN it.

```
LIST

10 PRINT "7 + 5=" ; 7 + 5
20 END
RUN

7 + 5= 12
```

If a PRINT statement contains more than one item (string or expression), the items must be separated by commas or semicolons.

remember... to get a copy of the program in the computer's memory, type LIST and press RETURN.

Tell the computer to

```
SCR
10 PRINT 7+5, 7-5, 7*5, 7/5
20 END
RUN

 12                 2                   35                  1.4
```

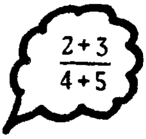
TO TELL THE COMPUTER TO ADD, USE +
TO TELL THE COMPUTER TO SUBTRACT, USE –
TO TELL THE COMPUTER TO MULTIPLY, USE *
TO TELL THE COMPUTER TO DIVIDE, USE /

To tell the computer to squeeze the answers more closely together, use semicolons instead of commas.

```
10 PRINT 7+5 ; 7-5 ; 7*5 ; 7/5
20 END
RUN

 12  2  35  1.4
```

Mixed operations? Try this one.

```
10 PRINT 2*3+4, 2*3+4*5, 2*3/4
20 END
RUN

 10               26               1.5
```



Use parentheses to group terms.

```
10 PRINT 2*(3+4), (2+3)*(4+5), (2+3)/(4+5)
20 END
RUN

 14               45               .555556
```

mistrakes

Do you occasionally make mistakes? We do, watch.

```
10 PTINT 2*3+4
```

We misspell PRINT.

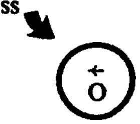
```
SYNTAX ERROR
```

The computer tells us we made a mistake.

The error message may be different on your computer. That's not the point. The point is, if we had noticed that we hit T when we meant to hit R, we could have corrected our mistake by using the back arrow.

BEWARE! THIS METHOD FOR CORRECTING MISTAKES MAY NOT WORK ON YOUR COMPUTER. IF IT DOESN'T ASK SOMEONE HOW TO MAKE CORRECTIONS.

The back arrow ← is on the same key as the letter 0. To type a back arrow, hold the SHIFT key down and press



```
SCR

10 PT←RINT 2*3+4
99 END
```

The back arrow (←) deletes the character that it points to.

```
LIST
```

LIST the program.

```
10 PRINT 2*3+4
99 END
```

The statement is O.K.

```
10 PRINT "MY HUMAN UNN←DERSTANS ←←DS ME"
99 END
```

Deletes 2nd N.   Deletes S and space.

```
LIST
10 PRINT "NY HUMAN UNDERSTANDS ME"
99 END
```

REMEMBER

A program is a set of statements. Each statement tells the computer to do some specific thing. So far, we have used only two types of statements, **PRINT** and **END**.

A statement begins with a line number. The computer obeys statements in line number order.

We space line numbers (10, 20, 30, etc.) so that we have room to insert new lines between existing line numbers. For example, we can insert up to nine new lines between Line 10 and Line 20.

You may choose line numbers arbitrarily and capriciously except for two things. A line number must be a positive integer between 1 and 9999, inclusive and the **END** statement must have the highest line number of any line in the program.

Type **SCR** to tell the computer to scratch (erase) the program in its memory. This is sort of like erasing a blackboard before you begin writing on it.

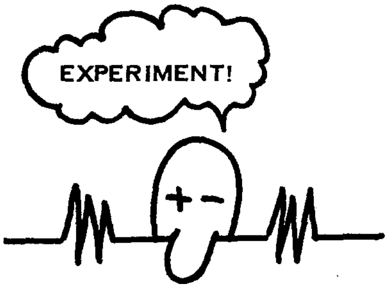
Type **RUN** to tell the computer to obey the program in its memory.

Type **LIST** to tell the computer to type the program in its memory on the TTY so you can read it.

When you type something, terminate each line by pressing the **RETURN** key. Nothing interesting will happen until you do.

To delete the last character typed, type a back arrow (←). To delete the last two characters, type two back arrows, to delete the last three characters, type three back arrows, and so on. (Remember, a space is a character too.)

And always remember . . .



SHORTHAND

The population of the U.S. is about 205 million people.

205 MILLION = 205000000

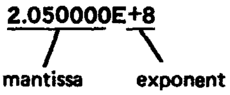
We asked *our*† computer to print the population of the U.S.

```
10 PRINT "POPULATION OF THE U.S. IS";205000000
99 END
RUN

POPULATION OF THE U.S. IS 2.050000E+08
```

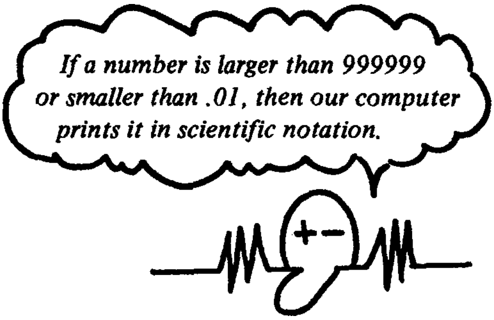
But we thought it was 205000000??!!

Our computer printed the population in *scientific notation.* (It really isn't especially scientific . . . it's just *called* scientific . . . some people call it *floating point.*)

Scientific notation is simply a shorthand way of expressing very large or very small numbers. In scientific notation a number is represented by a *mantissa* and an *exponent.*



The mantissa and the exponent are separated by the letter "E" . . . read on!



---

† *Your computer may do it somewhat differently.*

Here are some examples showing numbers written in good old everyday ordinary notation and again in scientific notation (well, scientific notation according to our computer).

one trillion

ordinary notation: **1 000 000 000 000**
scientific notation: **1.000000E+12**

volume of the Earth, in bushels

ordinary notation: **31 708 000 000 000 000 000 000**
scientific notation: **3.170800E+22**

For these two very large numbers, the exponents are *positive*.

speed of a snail in miles per second

ordinary notation: **.0000079**
scientific notation: **7.900000E–06**

mass of a hydrogen atom, in kilograms

ordinary notation: **.000 000 000 000 000 000 001 67**
scientific notation: **1.670000E–21**

For these two very small numbers, the exponents are *negative*.

Have you noticed? Our computer always prints the mantissa with 7 digits, one digit to the left of the point, 6 digits to the right.

$$\underbrace{3.170800}_{\uparrow \text{ 7 digits}}\text{E+22}$$

Also notice that the exponent is positive for large numbers and negative for small numbers.

3.170800E+22 ←—— exponent is positive (+22)

1.670000E–21 ←—— exponent is negative (–21)

Numbers printed in scientific notation can be converted to ordinary notation like this.

CASE 1. Exponent is positive.

(1) Write down the mantissa separately.

(2) Move the decimal point to the *right* the number of places specified by the exponent. If necessary, add zeros.

Example.

Computer prints	2.050000E+08
Write mantissa separately	2.050000
Move decimal point 8 places right	205000000.
	8 places (we had to add 2 zeros)

Ordinary notation: 205000000.

Again. Computer prints	3.170800E+22
Write mantissa separately	3.170800
Move decimal point 22 places right	31708000000000000000000.
	22 places (we had to add zeros)

Ordinary notation: 31708000000000000000000.

CASE 2. Exponent is negative.

(1) Write down mantissa separately.

(2) Move decimal point to the *left* the number of places specified by the exponent. If necessary, add zeros.

Example.

Computer prints	7.900000E-06
Write mantissa separately	7.900000
Move decimal point 6 places left	.000007900000
	6 places (we had to add 5 zeros)

Ordinary notation: .0000079

TOO MANY PEOPLE

At the end of 1970, the population of the earth was about 3.6 BILLION people.

☆ **3.6 BILLION = 3,600,000,000 = 3.6E9**

If the present growth rate persists, the population will double every 35 years. Suppose this actually happens . . . what will the population be in the year 2250?

$$\frac{2250 - 1970}{35} = \frac{280}{35} = 8 \text{ doublings}$$

We could do it like this.

```
10 PRINT 3.6E9*2*2*2*2*2*2*2*2
99 END
RUN
```

(8 doublings ... count them!)

```
9.216000E+11
```

← too many

How many people?

9.216E+11 = 921600000000 = 921.6 BILLION

A shorter way.

Do you remember? $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8$

In BASIC, we write 2^8 like this: 2↑8

```
10 PRINT 3.6E9*2↑8
99 END
RUN
```

Multiply 3.6E9 by 2^8

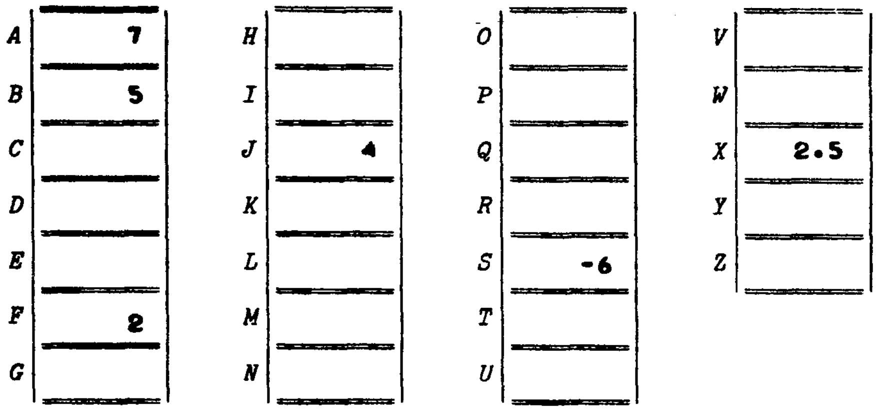
```
9.216000E+11
```

← Still too many people!

Remember ... to compute a *power* use ↑ N

BOXES

Deep down inside the computer there are 26 little boxes.



Each box can contain one number at any one time. We have already stored numbers in some of the boxes.

7 IS IN BOX A

5 IS IN BOX B

What number is in box F? ______ *In J?* ______

–6 is in box _______ *and 2.5 is in box* _______

O.K., using a *pencil*, put 8 into C. In other words, write the numeral "8" in the box labelled "C." Then do the following, carefully!

FIRST – *Put 12 into N.*

SECOND – *Put 27 into N. But wait! A box can hold only one number at a time . . . before you can enter 27 into N, you must first erase the 12 that you had previously entered.*

When the computer puts a number into a box, it *automatically* erases the previous content of the box.

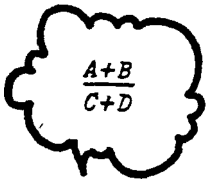
Tell it to the computer.

```
10 LET A = 7 <-------- PUT 7 INTO BOX A.
20 PRINT A   <-------- PRINT THE CONTENT OF BOX A.
99 END
RUN

 7
```

Another example.

```
10 LET A = 7
20 LET B = 5
30 PRINT A+B, A-B, A*B, A/B
99 END
RUN

 12            2              35             1.4
```

More practice? O.K.



```
10 LET A = 2
20 LET B = 3
30 LET C = 4
40 LET D = 5
50 PRINT A+B+C+D, A*B*C*D, A*(B+C), (A+B)/(C+D)
99 END
RUN

 14            120            14             .555556
```

We call **A, B, C, . . . , Z variables.** The number in box A is the **value of A,** the number in box B is the value of B, the number in C is the value of C, and so on. Without using the computer, complete each of the following RUNS as you think the computer would do it. Then use the computer to find out if you are correct.

```
10 LET A = 1
20 LET A = 2
30 PRINT A
99 END
RUN
```

____

```
10 LET A = 7
20 LET B = A
30 PRINT B
99 END
RUN
```

____

```
10 LET A = 1
20 PRINT A
30 LET A = 2
40 PRINT A
99 END
RUN
```

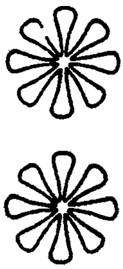
____

____

DIVISION OF LABOR

You and the computer have worked diligently. But you have done too much of the work, the computer too little.

A PROBLEM. In year zero, we start with a population of P people. The population increases by 1% each year. In N years, the population will be:



1% increase per year

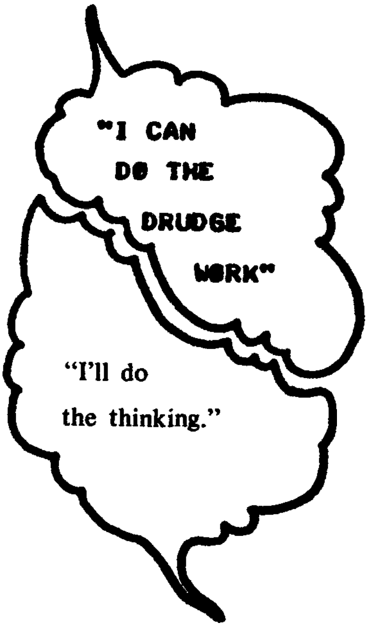
$$Q = P(1 + 1/100)^N \longleftarrow \text{N years}$$

initial population

population at the end of N years

If the growth rate is 2% per year, then

$$Q = P(1 + 2/100)^N$$



If the growth rate is 2.5% per year, then

$$Q = P(1 + 2.5/100)^N$$

And, if the growth rate is R% per year; then

$$Q = P(1 + R/100)^N$$

Remember . . .

P is the initial population.
R is the growth rate in per cent per year.
N is the number of years.
Q is the population in N years.

Let's write a BASIC program to compute and print the value of Q for given values of P, R and N.

You do it! First, type SCR and press the RETURN key. Then enter the program. Enter each statement as shown (and press the RETURN key after each statement, of course).

```
10 INPUT P
20 INPUT R
30 INPUT N
40 LET Q=P*(1+R/100)↑N
50 PRINT Q
99 END
```

Now type RUN and press RETURN. The computer types a question mark.

```
?
```

It wants something. It wants what the program tells it to want . . . INPUT P . . . do it. Enter 1000 as the initial population and press RETURN.

```
? 1000
?
```

Another question mark. Now what does it want? Of course, the value of R. Do it, enter 1 (for 1%) and press RETURN.

```
? 1000
? 1
?
```

Still another question mark. That's right. The computer wants the value of N. Let's try 20.

```
? 1000
? 1
? 20

 1220.19
```

It computes and prints the value of Q and stops. The program is still in the computer's memory. RUN it again. Use 1000 as the initial population, increase it 2% each year for 35 years.

```
RUN
? 1000
? 2
? 35

 1999.89
```

To **RUN** the program

Type **RUN** and press the **RETURN** key.

The computer types a question mark. Type the value of **P** and press the **RETURN** key.

The computer types a question mark. Type the value of **R** and press the **RETURN** key.

The computer types a question mark. Type in the value of **N** and press the **RETURN** key.

DO IT!

```
RUN

? 3.6E9             P = 3.6 billion people (Earth, 1970)
? 2                 R = 2%
? 31                N = 31 years

 6.651319E+09       Q = 6.7 billion people (Earth, 2001)
```

Your turn. Complete the following.

```
RUN

? 205E6             P = 205 million people (USA, 1970)
? 1.1               R = 1.1%
? 100               N = 100 years

________________
```

RUN the program again. You pick the values of P, R and N.

```
RUN

? —                 Your value of P.

?                   Your value of R.

?                   Your value of N.
```

*

Now that we have solved the problem one way, let's do it again using a slightly different approach.

the problem

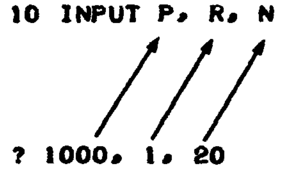
Compute Q for given values of P, R and N.

$$Q = P(1 + R/100)^N$$

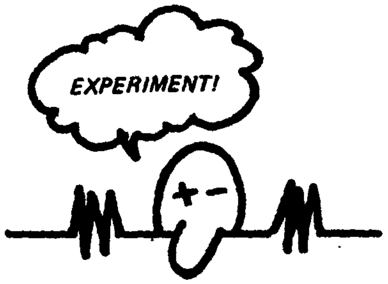
As usual, we want you to do it. Type as directed below.

```
SCR                                        First, SCRatch the old program.

10 INPUT P, R, N
20 LET Q = P*(1+R/100)^N  <=  Enter the new program
30 PRINT Q
99 END

RUN                                        Then RUN the new program.

? 1000, 1, 20                              Enter numbers for P, R, N.
 1220.19                                   Here is the answer.

RUN                                        RUN it again.

? 3.6E9, 2, 31                             Enter numbers for P, R, N.
 6.651319E+09                              Here is the answer.

? ___,___,___                              You RUN it, for your values of P, R, N.
 ______________         <------            Here is your answer.
```

In the above program there is only *one* INPUT statement, but it asks for input of *three* numbers, one number for each variable in the INPUT statement.



```
10 INPUT P, R, N

? 1000, 1, 20
```



Follow the Signs

Is the program on Page 18 still in the computer? If not, enter it.

Now we assume that the program on Page 18 is in the computer. Do not type SCR. Instead, type

```
40 GO TO 10
```

You have *added* a statement to the program in the computer. It tells the computer to GO TO Line 10. Let's look at the program as it now resides in the computer's memory.

type

```
LIST                               and press RETURN. The computer types the complete program.

10 INPUT P, R, N
20 LET Q = P*(1+R/100)↑N
30 PRINT Q
40 GO TO 10           <-------     Here is the statement you added, right where it should be, between
99 END                             Line 30 and Line 99.

RUN

? 1000, 1, 20                      Enter values.
 1220.19                           Computer prints answer and goes around.
? 1000, 2, 35                      Enter values.
 1999.890                          Computer prints answer and goes around.
? 3.6E9, 2, 100                    Enter values.
 2.608073E+10                      Computer prints answer and goes around.

? ____________                     YOUR TURN. Enter values.

  ____________                     Computer prints answer and goes around.
```

? But how do we get out of this??

TYPE STOP AND PRESS RETURN.
IF THAT DOESN'T WORK, HOLD THE
CTRL KEY DOWN AND PRESS THE C
KEY. THEN LET GO AND PRESS
RETURN. IF THAT DOESN'T WORK,
PRESS ESC OR ALT MODE. IF
THAT DOESN'T WORK, YELL FOR
HELP!

Good luck!

READ and DATA READ and DATA READ and DATA READ and DATA

Let's start with the program on Page 19 and make a couple of changes. Change INPUT to READ . Change the PRINT statement. Add three DATA statements. Each DATA statement contains one set of data . . . values of P, R, N.

```
10 READ P, R, N
30 PRINT P, R, N, Q

90 DATA 1000, 1, 20
91 DATA 1000,2,35
92 DATA 3.6E9, 2, 100
```

```
LIST

10 READ P, R, N
20 LET Q = P*(1+R/100)↑N
30 PRINT P, R, N, Q
40 GO TO 10
90 DATA 1000, 1, 20
91 DATA 1000,2,35
92 DATA 3.6E9, 2, 100
99 END
```

Here is a change.

Here is a change.

And three added statements.

```
RUN

 1000             1            20             1220.19
 1000             2            35             1999.890
 3.600000E+9      2            100            2.608072E+10

OUT OF DATA IN LINE 10
```

This means what it says. The computer has used all the data in the DATA statement.

BEWARE!____*the message may be different on your computer.*

Now it's your turn. Add the following statement,

```
5 PRINT " P"," R"," N"," Q"
```

and RUN the program. The results should look like this:

```
 P                R            N              Q

 1000             1            20             1220.19
 1000             2            35             1999.889
 3.600000E+09     2            100            2.608072E+10

OUT OF DATA IN LINE 10
```

We used three DATA statements, each with one set of values for P, R and N. But we could have put all the data in one DATA statement. Try it.

type

```
90 DATA 1000, 1, 20, 1000, 2, 35, 3.6E9, 2, 100
```

Wait! Since you have put all the data in Line 90, you no longer need Lines 91 and 92. Let's delete them.

```
91                    IF YOU TYPE A LINE NUMBER AND PRESS
92                    RETURN, THE COMPUTER ERASES THAT LINE,
                      BUT ONLY THAT LINE, FROM ITS MEMORY.
```

list & run

```
LIST
5 PRINT " P"," R"," N"," Q"
10 READ P,R,N
20 LET Q=P*(1+R/100)↑N
30 PRINT P,R,N,Q
40 GO TO 10
90 DATA 1000,1,20,1000,2,35,3.600000E+09,2,100
99 END
RUN

 P               R               N               Q
 1000            1               20              1220.19
 1000            2               35              1999.889
 3.600000E+09    2               100             2.608072E+10

OUT OF DATA IN LINE 10
```

READ tells the computer to read numbers from a DATA statement, one number for each variable in the READ statement. Use as many DATA statements as you wish to hold the data. When the computer uses all the data in a DATA statement, it automatically moves to the next DATA statement. If it can't find any more data, it types a message such as OUT OF DATA.

Next a program to print a table showing the value of Q versus the value of N for fixed values of P and R.

```
100 READ P,R
110 PRINT "INITIAL POPULATION IS";P;"PEOPLE"
120 PRINT "GROWTH RATE IS";R;"%"
130 PRINT

200 PRINT " N","POPULATION"
210 PRINT

300 READ N
310 LET Q=P*(1+R/100)↑N
320 PRINT N,Q
330 GO TO 300

900 DATA 100000,2
910 DATA 10,20,30,31,35,70,100
999 END
```

Most of the work is done here.

```
RUN

INITIAL POPULATION IS 100000 PEOPLE
GROWTH RATE IS 2 %

 N             POPULATION

 10             121899.4
 20             148594.7
 30             181136.1
 31             184758.9
 35             199988.9
 70             399955.7
 100            724464.4

OUT OF DATA AT LINE 300
```

These results are brought to you by Lines 110 and 120.

Courtesy of Line 200.

Thanks to Lines 300 – 330

CHECK OUT THE PROGRAM:

✓ The values of P and R are in Line 900. There is *one* value of P and *one* value of R.

✓ The values of N are in Line 910. There are *seven* values of N.

Your turn. Change the data. Use data for Earth, 1970, and values of N as follows:

P = 3.6E9

R = 2

N = 10,20,30,31,35,70,100,200,300

DEMOGRAPHY

Demography?

> **de·mog·ra·phy** [*Gr. demos, the people + -GRAPHY*] ***the statistical science dealing with the distribution, density, vital statistics, etc. of populations.***

A good source of demographic data:

Me
My address

POPULATION REFERENCE
BUREAU, INC.
1755 Massachusetts Ave., N.W.
Washington, D.C. 20036

Their 1970 WORLD POPULATION DATA SHEET gives data on 142 countries summarized by geographical regions. We have copied a small amount of data from the data sheet.

REGION	POPULATION [millions]	GROWTH RATE [% per year]
AFRICA	344	2.6
ASIA	2056	2.3
NORTH AMERICA	228	1.1
LATIN AMERICA	283	2.9
EUROPE	462	.8
U.S.S.R.	243	1.0
OCEANIA	19	2.0
WORLD	3635	2.0

Get acquainted with the data. You will see a lot of it from now on!

O.K., we now have a *data base* (set of data) consisting of 1970 population and growth rate for eight regions, 16 numbers in all.

First, a simple program to read the data and print the table. Our program includes REMARK statements which (we hope) make the program easier for *people* to read and understand. The computer ignores REMARK statements.

```
100 REMARK***PRINT THE HEADING
110 PRINT "1970 WORLD POPULATION DATA"
120 PRINT
130 PRINT "POPULATION","GROWTH RATE"
140 PRINT

200 REMARK***READ AND PRINT NUMERICAL DATA
210 READ P, R
220 PRINT P, R
230 GO TO 210

900 REMARK***DATA BASE
910 DATA 344, 2.6
920 DATA 2056, 2.3
930 DATA 228, 1.1
940 DATA 283, 2.9
950 DATA 462, .8
960 DATA 243, 1.0
970 DATA 19, 2.0
980 DATA 3635, 2.0

999 END

RUN
1970 WORLD POPULATION DATA

POPULATION     GROWTH RATE

 344            2.6
 2056           2.3
 228            1.1
 283            2.9
 462            .8
 243            1
 19             2
 3635           2

OUT OF DATA AT LINE 210
```

population given in
1,000,000's *

In order to save space, let's put more data in each DATA statement and use fewer DATA statements. You do it. Make the following changes:

```
910 DATA 344,2.6,2056,2.3
920 DATA 228,1.1,283,2.9,462,.8
930 DATA 243, 1.0,19, 2.0,3632,2.0
```

PUT ALL THE DATA
IN LINES 910–930.

If the program is still in the computer, enter the changes noted above:

- ✓ **REPLACE** the old Lines 910, 920, 930 with the ones above.
- ✓ **DELETE** Lines 940 to 980. (If you've forgotten how, see page 21.)
- ✓ **RUN** the new program.

The results should be the same as the results produced by the original program.

Your Turn ___

Modify the program so that, for each region, the computer computes and prints a third column: the population in the year 2001. In other words, a run of your program might look like the following: (Don't change the data base!)

```
RUN

1970 WORLD POPULATION DATA

POPULATION        GROWTH RATE       POP. IN 2001

 344               2.6               762.3011
 2056              2.3               4160.711
 228               1.1               320.052
 283               2.9               686.5356
 462                .8               591.4503
 243               1                 330.8025
 19                2                 35.10418
 3635              2                 6710.442 ← World Population in 2001.
```

KEEP EXPERIMENTING! Change the program again so that the third column gives the population for the year 1984. Or for the year 2500. Or . . .

* * *

By the way, in examining the above results, we noticed an odd thing. The first seven population figures in the column headed POP. IN 2001 should add up to the eighth figure in the column (World population in 2001), but they don't. Why not?

BEWARE mathematical models

We made an assumption and developed a mathematical model of population growth.

ASSUMPTION: The increase in population each year is a percentage of the population at the beginning of the year. The percent increase, which we call **R**, does not change from year to year.

MATH MODEL: If the initial population is **P** and the (constant) rate of increase is **R%** per year, then the population **Q** in **N** years is

$$Q = P(1 + R/100)^N$$

QUESTIONS

Does our model really resemble real life?

Can we use it to predict the future population of the U.S.? Of the Earth?

How far into the future can we expect our model to provide reasonably accurate predictions? 10 years? 100 years? 1000 years?

The above questions lead to more questions.

Is the rate of increase **(R)** really constant or is it increasing or decreasing?

Is the rate the same for different regions of the Earth (e.g., North America, Asia, and so on)?

Can we look more deeply into the mechanisms of population growth (birth rate, death rate, migration, life expectancy, fertility, and so on)?

Where can we get more information?

The last question we can answer. ➡

Population Reference Bureau, Inc.
1755 Massachusetts Avenue N.W.
Washington, D.C. 20036

$$Q = P(1 + R/100)^n$$

SORCERER'S APPRENTICE

Do you know the story about the Sorcerer's Apprentice? While the Sorcerer was gone, the apprentice instructed the magic broom to fetch water from the well. The broom complied and began carrying water, more water, more water . . . the apprentice had forgotten how to tell the broom to stop.

The following program makes the computer behave like the Sorcerer's broom. Once you set it in motion, it will start printing, printing, printing, . . . you, the apprentice, must know how to stop it!

Before typing the program, find the BREAK key. It is on the righthand side of the keyboard.

Now, enter the program.

```
10 LET N = 1
20 PRINT N
30 LET N = N+1
40 GO TO 20
99 END
```

BEFORE TYPING RUN, READ THIS:

To STOP the computer,

Press BREAK for 1 second.

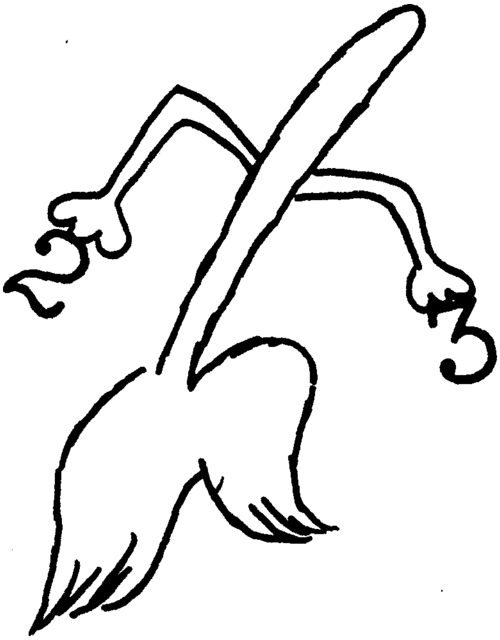
If that doesn't work, press the S key.

If that doesn't work, try ESC or ALTMODE.

If that doesn't work, yell for help!

```
RUN

 1
 2
 3
 4
 5
 6
 7
 8
 •
 •
 •
```

and so on forever unless you stop the computer!



Let's follow along as the computer RUNs the program on the preceding page. *Follow the arrows.*

GOTO...

```
START HERE

10 LET N = 1

20 PRINT N

30 LET N=N+1

40 GO TO 20

99 END
```

This is a loop. It is a "forever" loop.
It goes on and on and on and on and on and on and on and on and on and on ...
until someone intervenes.

Another view. Below is a TRACE of the program. *(Trace? Sure! A TRACE traces the path the computer takes through the program it is working on, and shows what values are assigned to the variables at any step in the program. Ain't it obvious?)* In the column marked N we show the value of N *after* the statement on the same line has been carried out by the computer.

STATEMENT	N	REMARKS
10 LET N=1	1	
20 PRINT N	1	Print the value of N.
30 LET N=N+1	2	Increase N by 1 *(add 1 to the old value of N).*
40 GO TO 20	2	Go to beginning of loop.
20 PRINT N	2	Print the value of N.
30 LET N=N+1	3	Increase N by 1.
40 GO TO 20	3	Go to beginning of loop.
20 PRINT N	3	Print the value of N.
30 LET N=N+1	4	Increase N by 1.
40 GO TO 20	4	Go to beginning of loop.

THE SORCERER RETURNS!

Here is our Sorcerer's Apprentice program again . . . but we have made one small change.

```
10 LET N=1
20 PRINT N
30 LET N=N+1

40 IF N<=3 THEN 20

99 END

RUN

 1
 2
 3

READY.
```

Here is the change. We use an IF ... THEN statement instead of a GO TO statement.

Read on!

The computer counts to 3, types READY and stops.

YOUR COMPUTER MAY TYPE D0NE OR SOME OTHER MESSAGE, OR PERHAPS, IT WILL JUST SIMPLY STOP.

This statement `IF N<=3 THEN 20`

tells the computer:

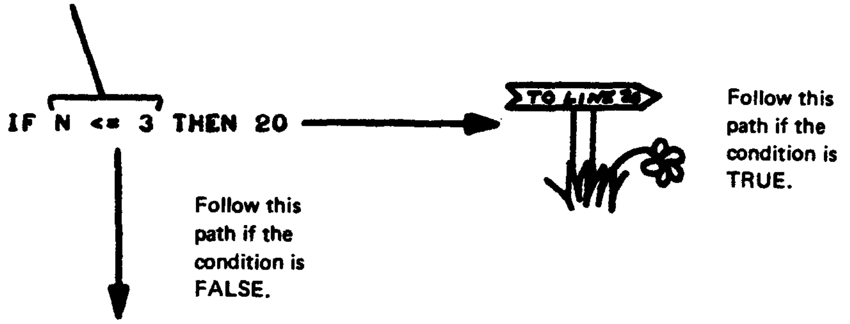
```
IF THE VALUE 0F N IS LESS THAN
0R EQUAL T0 3 THEN G0 T0 LINE 20.
```

If the value of N is **NOT** less than or equal to 3, the computer goes on to Line 99 (the next line in regular line number sequence). And, since Line 99 is an END statement, the computer stops.

IF...THEN...

Follow the arrows. Read the road signs.

This is a *condition.*

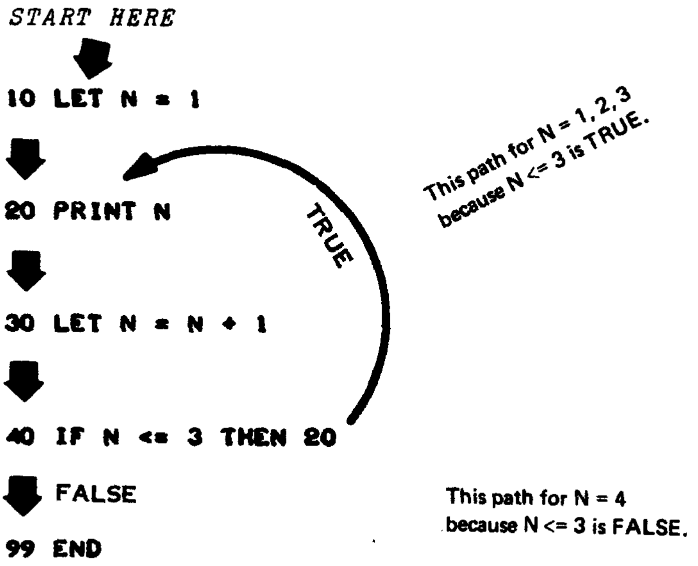


```
IF N <= 3 THEN 20
```

Follow this path if the condition is TRUE.

Follow this path if the condition is FALSE.

Keep following.



```
10 LET N = 1
20 PRINT N
30 LET N = N + 1
40 IF N <= 3 THEN 20
99 END
```

This path for N = 1, 2, 3 because N <= 3 is TRUE.

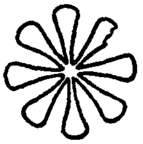
This path for N = 4 because N <= 3 is FALSE.

There is always another way. Here are two more programs to "count to 3." RUN them.

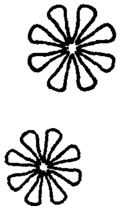
```
10 LET N = 1
20 PRINT N
30 IF N = 3 THEN 99
40 LET N = N + 1
50 GO TO 20
99 END
```

```
10 LET N = 1
20 IF N > 3 THEN 99
30 PRINT N
40 LET N = N + 1
50 GO TO 20
99 END
```

world of IF



Another look at the IF statement.



GENERAL FORM: **IF** condition **THEN** line number

EXAMPLE: **IF** **K>3** **THEN** **99**

The condition **K>3** is true for some values of **K**, false for other values.

Suppose K = 4.	Then K>3 is TRUE. The computer will go to Line 99.
Suppose K = 1.	Then K>3 is FALSE. The computer will go to the line number that is next higher than the line number of the IF statement.

What happens for K = 2? K = 3? __________ __________

The condition in an IF statement is usually a math *relation* between two BASIC expressions. The permissible relations are shown in the table below.

RELATION	MATH SYMBOL	BASIC SYMBOL
equal to	=	=
less than	<	<
greater than	>	>
less than or equal to	≤	<=
greater than or equal to	≥	>=
not equal to	≠	<>

Do you understand all you know about the IF statement? Find out . . . *predict* the results printed by the computer under control of each of the following programs. Then RUN them to find out if you are correct.

```
10 LET N = 0
20 PRINT N
30 LET N = N + 10
40 IF N<=100 THEN 20
99 END
```

```
10 LET C = 10
20 PRINT C
30 IF C=0 THEN 99
40 LET C = C-1
50 GO TO 20
99 END
```

Let's put IF to work. The following program directs the computer to generate and print a table of

$$Q = P(1 + R/100)^N$$

for equally spaced values of N ($N = 0, 10, 20, \ldots, 100$).

```
10 PRINT "INITIAL POPULATION";
15 INPUT P
```

Don't forget the semicolon. Why? See Page 33.

```
20 PRINT "GROWTH RATE";
25 INPUT R

30 PRINT
40 PRINT " N","POPULATION"
45 PRINT
```

Print the heading.

```
50 LET N=0
```

Initial value of N is zero.

```
60 LET Q=P*(1+R/100)↑N
70 PRINT N,Q
80 LET N=N+10
90 IF N<=100 THEN 60

99 END
```

This is a loop. Line 60–90 are carried out for N = 0, 10, 20, . . . , 100.

```
RUN
```

RUN it for USA, 1970.

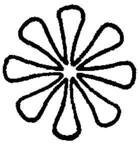
```
INITIAL POPULATION?205
GROWTH RATE?1
```

Given in millions of people.
1% growth rate.

```
 N              POPULATION

 0               205
 10              226.4475
 20              250.1389
 30              276.309
 40              305.217
 50              337.1494
 60              372.4226
 70              411.3862
 80              454.4263
 90              501.9693
 100             554.4863

READY.
```

Since the initial population was given in millions of people, the results are also in millions of people. For N = 50, the population is

337.1494 million people.



The statements:

```
10 PRINT "INITIAL POPULATION";
15 INPUT P
```

Tell the computer to type:

```
INITIAL POPULATION ?
```

Line 10 tells the computer to type INITIAL POPULATION. The semicolon at the end of Line 10 says "don't RETURN the carriage to the left margin." Line 15 tells the computer to type a question mark and wait.

What would happen if we omitted the semicolon at the end of Line 10? Try it . . . find out for yourself.

What would happen if we use a comma instead of a semicolon? Try it . . . find out.

* * *

Let's make a small change in the program.

```
70 PRINT N,INT(Q+.5)
```

What is INT? Accept it for now. RUN the modified program. The RUN should look like this:

```
RUN

INITIAL POPULATION?205
GROWTH RATE?1

 N                   POPULATION

 0                    205
 10                   226
 20                   250
 30                   276
 40                   305
 50                   337
 60                   372
 70                   411
 80                   454
 90                   502
 100                  554

READY.
```

Compare the POPULATION results with the results on Page 32.

These results are all *rounded* to the nearest whole number.

INT

The INT function has the general form

INT(e)

where *e* is any BASIC expression. The INT function tells the computer to evaluate the expression and then compute the greatest integer that is less than or equal to the value of the expression.

Examples.

INT(2) = 2
INT(0) = 0
INT(–3) = –3
INT(3.99) = 3
INT(.01) = 0
INT(.999) = 0
INT(–.01) = –1
INT(–3.14) = –4
INT(25/2) = 12

More examples? Gather your own. RUN the following program.

xXX *Be creative . . . choose both plain and fancy x's.* XXX

```
10 PRINT "X=";
20 INPUT X
30 PRINT
40 PRINT"INT(X)=";INT(X),"INT(X+.5)=";INT(X+.5)
50 PRINT
60 GO TO 10
99 END
```

* * *

(1) INT(6.7) = __________
(2) INT(6.7 + .5) = __________
(3) INT(6.3) = __________
(4) INT(6.3 + .5) = __________
(5) INT(6.5) = __________
(6) INT(6.5 + .5) = __________
(7) INT(–3.9) = ________
(8) INT(–3.9 + .5) = __________
(9) INT(–3.4) = ________
(10) INT(–3.4 + .5) = _________

* * *

RACE TO OBLIVION

Here is our World Population table again. Population is given in millions of people.

REGION	POPULATION	RATE OF INCREASE
AFRICA	344	2.6%
ASIA	2056	2.3
NORTHERN AMERICA	228	1.1
LATIN AMERICA	283	2.9
EUROPE	462	0.8
U.S.S.R.	243	1.0
OCEANIA	19	2.0
WORLD	3635	2.0

The fastest growing region is Latin America and the slowest growing region is Europe. In 1970, the population of Europe was considerably more than the population of Latin America.

ASSUME: **The growth rates for Europe and Latin America will remain constant.**

QUESTION: **In what year will the population of Latin America become greater than the population of Europe?**

Try this program.

```
10 LET N=1
20 LET E=462*(1+.8/100)↑N           E for Europe
30 LET L=283*(1+2.9/100)↑N          L for Latin America
40 IF L>E THEN 70
50 LET N=N+1
60 GO TO 20
70 PRINT "THE YEAR IS";1970+N
99 END
RUN

THE YEAR IS 1994

READY.
```

If the assumption is correct, the population of Latin America becomes greater than the population of Europe in 1994.

A more general program. We enter the 1970 population and growth rate for each population. The computer computes and prints the year in which the *second* population overtakes the *first* population.

```
100 PRINT "FIRST POPULATION";
110 INPUT P1
120 PRINT "GROWTH RATE";
130 INPUT R1
140 PRINT
150 PRINT "SECOND POPULATION";
160 INPUT P2
170 PRINT "GROWTH RATE";
180 INPUT R2
190 PRINT
200 LET N=1
210 LET Q1=P1*(1+R1/100)↑N
220 LET Q2=P2*(1+R2/100)↑N
230 IF Q2>Q1 THEN 300
240 LET N=N+1
250 GO TO 210
300 PRINT "THE YEAR IS";1970+N
310 PRINT
320 GO TO 100
999 END
RUN
```

A BASIC VARIABLE MAY BE A LETTER FOLLOWED BY A DIGIT.

```
FIRST POPULATION?462
GROWTH RATE?.8
```

← Europe

```
SECOND POPULATION?283
GROWTH RATE?2.9
```

← Latin America

```
THE YEAR IS 1994

FIRST POPULATION?
```

Your turn. But wait! Read on.

ASSUMPTIONS. We assume that the first population is greater than the second population in 1970 but that the second population is growing more rapidly. Mathematically, we say that

$$P1 > P2 \quad \text{and} \quad R2 > R1$$

Try some data that violates one or both assumptions.

Beware! You may have to stop the comouter if you enter data for which the second population can't catch the first population.

We want to sneak in a new idea . . . checking for valid data. First, answer a few questions.

1. Will Northern America ever catch up with Latin America?
2. What happens if we enter Oceania as the first population and Asia as the second population?
3. Will Africa ever catch up with Latin America?

Remember, our assumptions are P1 > P2 **and** R2 > R1. **Let's add some statements to the program to check the incoming data and reject data that violates one of the assumptions. The data violate the assumptions if**

P1 <= P2   or   R2 <= R1

Add the following statements.

```
181 IF P1<=P2 THEN 184
182 IF R2<=R1 THEN 184
183 GO TO 190
184 PRINT "BAD DATA. TRY AGAIN."
185 PRINT
186 GO TO 100
```

LIST the modified program. RUN it. Use several sets of data. After all, there are 56 different ways to select a FIRST POPULATION and a different SECOND POPULATION from the eight regions (including the World total) shown in the table.

One more thing. Will the population of Latin America ever become greater than the population of the entire World? Try it on the computer. Use data for World as FIRST POPULATION and data for Latin America as SECOND POPULATION.

BEWARE MATHEMATICAL MODELS

Your Turn

Modify the program of Pages 36 and 37 so that results are printed as indicated below.

```
RUN

FIRST POPULATION?462
GROWTH RATE?.8

SECOND POPULATION?283
GROWTH RATE?2.9

THE YEAR IS 1994
FIRST POPULATION IS NOW 559
SECOND POPULATION IS NOW 562

FIRST POPULATION? and so on.
```

Modify the program so that the values of *P1*, *R1*, *P2*, and *R2* are entered by means of READ and DATA statements. In fact, use the following DATA statements. (Add line numbers.)

```
DATA 462, .8, 283, 2.9
DATA 462, .8, 344, 2.6        GOOD DATA
DATA 462, .8, 243, 1
DATA 344, 2.6, 283, 2.9

DATA 283, 2.9, 228, 1.1
DATA 228, 1.1, 283, 2.9       BAD DATA
DATA 283, 2.9, 462, .8

      ↑    ↑    ↑    ↑
      P1   R1   P2   R2
```

For valid data, numerical results should be printed under the following headings. We show results corresponding to the data in the first (top) DATA statement above.

```
FIRST POP.     SECOND POP.    CATCHUP        NEW FIRST      NEW SECOND
 1970           1970           YEAR          POPULATION     POPULATION

 462            283            1994           559            562
```

Does this program look familiar? (It should . . .)

```
10 LET N = 1
20 PRINT N
30 LET N = N + 1
40 IF N<=3 THEN 20
99 END

RUN

 1
 2
 3

READY
```

You first saw this program on Page 29. It tells the computer to count to 3, and then stop.

A NEW IDEA . . . AND ANOTHER WAY TO COUNT TO 3

```
10 FOR N=1 TO 3
20 PRINT N
30 NEXT N
99 END

RUN

 1
 2
 3

READY.
```

This is a FOR–NEXT loop.

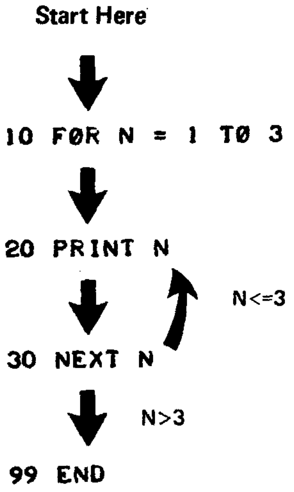
Count to 3.

Then stop.

a FOR-NEXT loop

- begins with a FOR statement
- ends with a NEXT statement
- usually includes one or more statements between the FOR and NEXT statements

How does the FOR-NEXT loop work? Follow the ➡ ➡ ➡



```
10 FOR N = 1 TO 3
20 PRINT N
30 NEXT N
99 END
```

In Line 10, N is set equal to one.

In Line 30, N is increased by one.

As you can see, each time the computer comes to the NEXT N statement, it increases the value of N by one, and checks the new value against the limit for N. In this case, the limit is 3, because the FOR statement reads: FOR N = 1 TO 3. When the value of N is greater than 3, the computer continues on to the next statement after the NEXT statement. (Got that?)

more examples

```
10 FOR N = 0 TO 3
20 PRINT N
30 NEXT N
99 END
RUN

 0
 1
 2
 3

READY
```

```
10 FOR N = 2 TO 7
20 PRINT N
30 NEXT N
99 END
RUN

 2
 3
 4
 5
 6
 7

READY
```

Got the idea? Then try these. *Without using the computer,* complete each of the following by filling in the blanks.

```
10 FOR N = 10 TO 13
20 PRINT N
30 NEXT N
99 END
RUN
____________

____________

____________

____________
READY
```

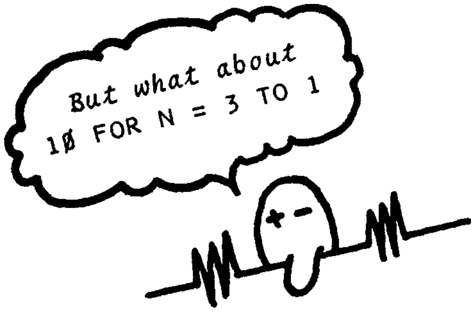
```
10 FOR N = -1 TO 1
20 PRINT N
30 NEXT N
99 END
RUN
__________

__________

__________
READY
```

Have you noticed that both programs are the same except for Line 10? Now check your work above by running the programs on the computer. Do it now.

And then......

Then, experiment! Check out each of the following FOR statements. Remember, you need only change Line 10.

```
10 FOR N = 5 TO 5

10 FOR N = 1.5 TO 6.5

10 FOR N = 1.25 TO 5.25

10 FOR N = 1.25 TO 5

10 FOR N = 1 TO 5.25

10 FOR N = 1 TO 2*3

10 FOR N = 2*3 TO 4*5

10 FOR N = 1/2 TO 17/2
```



Don't get the idea that you may only use "N" as the variable in a FOR-NEXT loop. Look for the FOR-NEXT loop in program on the next page.

To get to
the next page,
place your thumb here
and your forefinger on
the other side of . . .

Next . . . a program to print the data from the 1970 WORLD POPULATION SHEET.

```
100 REMARK***PRINT THE HEADING
110 PRINT "1970 WORLD POPULATION DATA"
120 PRINT
130 PRINT "REGION NO.","POPULATION","GROWTH RATE"
140 PRINT

200 REMARK***READ AND PRINT NUMERICAL DATA
210 FOR K=1 TO 8
220 READ P,R
230 PRINT K,P,R
240 NEXT K

900 REMARK***DATA BASE
910 DATA 344,2.6,2056,2.3
920 DATA 228,1.1,283,2.9
930 DATA 462,.8,243,1
940 DATA 19,2,3635,2

999 END

RUN

1970 WORLD POPULATION DATA

REGION NO.     POPULATION     GROWTH RATE

 1               344           2.6
 2               2056          2.3
 3               228           1.1
 4               283           2.9
 5               462           .8
 6               243           1
 7               19            2
 8               3635          2

READY.
```

We will refer to the above program again. Therefore, we suggest that you learn how to "dump" the program on paper tape so that when you want to enter it again, you can do so quickly, using the paper tape reader. (Saves a lot of typing and a lot of terminal time!!!)

The EASY way:	*Ask someone to show you how.*
The HARD way:	*Dig it out of the reference manual or operating manual for BASIC on the computer system that you are using.*

COUNT TO N

The following program directs the computer to *count to N*, where the value of *N* is supplied in response to an INPUT statement.

```
10 PRINT "N=";
20 INPUT N

30 FOR K = 1 TO N
40 PRINT K
50 NEXT K

60 PRINT
70 GO TO 10
99 END
```

```
RUN

N= ?7
```

We enter 7.

```
 1
 2
 3
 4
 5
 6
 7
```

The computer counts to 7.

```
N= ?3
```

We enter 3.

```
 1
 2
 3
```

The computer counts to 3.

```
N= ?
```

and so on...

* * *

Change Line 40 as follows and RUN the program again.

```
40 PRINT N-K+1
```

EXPERIMENT!

Now we want to use the program on Page 42 again. *Did you dump it onto paper tape?* (That is, did you punch a paper tape copy of the program after you typed it in the first time?) If so, read the program into the computer through the paper tape reader. Other wise, type it in again ... slowly, tediously, by hand!

◆◆◆

Now that you have entered the program, make the following changes:

```
205 READ N
210 FOR K = 1 TO N
905 DATA 8
```

Then LIST the modified program

```
LIST
100 REMARK***PRINT THE HEADING
110 PRINT "1970 WORLD POPULATION DATA"
120 PRINT
130 PRINT "REGION NO.","POPULATION","GROWTH RATE"
140 PRINT
200 REMARK***READ AND PRINT NUMERICAL DATA
205 READ N
210 FOR K=1 TO N
220 READ P,R
230 PRINT K,P,R
240 NEXT K
900 REMARK***DATA BASE
905 DATA 8
910 DATA 344,2.6,2056,2.3
920 DATA 228,1.1,283,2.9
930 DATA 462,.8,243,1
940 DATA 19,2,3635,2
999 END
```

LINE 905 CONTAINS
THE VALUE OF M.
LINES 910-940 CONTAIN
M SETS OF DATA (P AND R)

Now RUN the modified program. The results should be exactly the same as the results in the RUN on Page 42.

You ask (and well you might): *If the results are the same, why did we bother?*

We answer: **Read on!**

And here is your very own data base to play with.

COUNTRY	K	POPULATION†	GROWTH RATE
AUSTRALIA	1	12.5	1.9
BRAZIL	2	93.0	2.8
CHINA	3	759.6	1.8
COSTA RICA	4	1.8	3.8
GERMANY, EAST	5	16.2	0.3
GERMANY, WEST	6	58.6	0.6
INDIA	7	554.6	2.6
JAPAN	8	103.5	1.1
MALTA	9	0.3	−0.8
MEXICO	10	50.7	3.4
NIGERIA	11	55.1	2.6
PHILIPPINES	12	38.1	3.4
U.S.S.R.	13	243.6	1.0
U.A.R.	14	33.9	0.5
UNITED KINGDOM	15	56.0	2.8
U.S.A.	16	205.2	1.0

† in millions of people, rounded to the nearest 10th of a million

How would you modify the program on Page 42 to use the above data base? (Go ahead and try such a modification if you wish.)

How would you modify the program on Page 44 to use the above data base?

PLEASE DO IT – you need to change *only* Lines 905, 910, 920, 930 and 940. If necessary, add additional DATA statements.

RUN the program after you have modified it. The results should look like this:

```
RUN

1970 WORLD POPULATION DATA

REGION NO.     POPULATION    GROWTH RATE

 1              12.5          1.9
 2              93            2.8
 3              759.6         1.8
```

et cetera

DO I ALWAYS HAVE TO STEP BY 1?

"What?"

"I said, do I always have to step by 1?"

"I thought you'd never ask. My Cogent answer: No. Try these on your friendly computer."

```
10 FOR K=1 TO 9 STEP 2
20 PRINT K
30 NEXT K
99 END
RUN

 1
 3
 5
 7
 9

READY.
```

K is stepped by 2.

```
10 FOR K=0 TO 10 STEP 3
20 PRINT K
30 NEXT K
99 END
RUN

 0
 3
 6
 9

READY.
```

K is stepped by 3.

Have you experimented?

****

We can even step "backwards." RUN this one.

****

```
10 FOR K=10 TO 0 STEP -1
20 PRINT K
30 NEXT K
99 END
```

EXPERIMENT
try these—

```
10 FOR K=1 TO 2 STEP .25
10 FOR K=0 TO .5 STEP .1
```

THE HANDY-DANDY SUPER-VERSATILE FOR-NEXT LOOP

There are two general forms of the FOR statement. Here is the first general form, and some examples.

FOR	*[variable]*	=	*[expression]*	**TO**	*[expression]*
FOR	N	=	1	TO	3
FOR	K	=	1	TO	N
FOR	X	=	A	TO	B
FOR	J	=	1	TO	M – 1
FOR	C	=	D/2	TO	D

In other words:

FOR ______	=	______	**TO**	______
any BASIC variable		*any BASIC expression*		*any BASIC expression*

* * *

The second general form is:

FOR	*[variable]*	=	*[expression]*	**TO**	*[expression]*	**STEP**	*[expression]*
FOR	K	=	1	TO	9	STEP	2
FOR	N	=	0	TO	100	STEP	10
FOR	X	=	A	TO	B	STEP	H
FOR	C	=	N+1	TO	2*M	STEP	K/2

In other words:

FOR	______	= ______	**TO**	______	**STEP**	______
	any BASIC variable	*any BASIC expression*		*any BASIC expression*		*any BASIC expression*

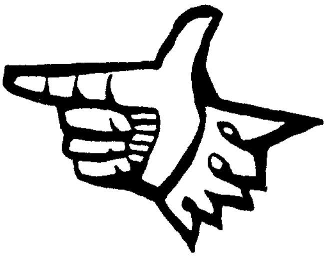
Back on Page 30 (*if you'll cast your mind back*) there is a program to print a table of:

$$Q = P(1 + R/100)^N$$

Here is a more general program.

```
100 REMARK***REQUEST DATA AND PRINT HEADING
110 PRINT "INITIAL POPULATION";
115 INPUT P
120 PRINT "GROWTH RATE";
125 INPUT R
130 PRINT "INITIAL VALUE OF N";
135 INPUT A
140 PRINT "FINAL VALUE OF N";
145 INPUT B
150 PRINT "STEP SIZE";
155 INPUT H
160 PRINT
170 PRINT " N","POPULATION"
180 PRINT

200 REMARK***COMPUTE AND PRINT TABLE
210 FOR N=A TO B STEP H
220 LET Q=P*(1+R/100)↑N
230 PRINT N,INT(Q+.5)
240 NEXT N

999 END

RUN

INITIAL POPULATION?205
GROWTH RATE?1
INITIAL VALUE OF N?0
FINAL VALUE OF N?100
STEP SIZE?10

 N              POPULATION

 0               205
 10              226
 20              250
 30              276
 40              305
 50              337
 60              372
 70              411
 80              454
 90              502
 100             554

READY.
```

For U.S.A., 1970 (in millions of people).



Results are rounded to the nearest million.

Compare with page 32.

RUN it again for input data of your choice.

SUBSCRIPTED VARIABLES

Until now, we have used only *simple* BASIC variables. A simple variable consists of a letter (any letter A to Z) or a letter followed by a single digit (any digit 0 to 9).

For example, the following are simple variables:

P R K P1 P2

Now we want to introduce a new type of variable, called a

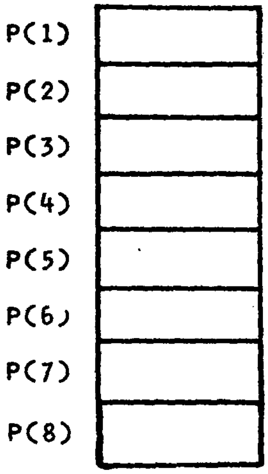
Subscripted variable

Subscripted variable: **P(5)**
Subscript: ↗

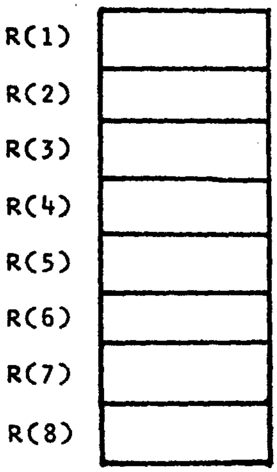
Say it like this: "P sub 5"

A subscripted variable names a location inside the computer; you can think of it as a box, a place to store a number.

EIGHT SUBSCRIPTED VARIABLES



EIGHT MORE SUBSCRIPTED VARIABLES



KNOW THIS:

P, P1 and P(1) are three *distinct* variables. All three can appear in the same program. They may confuse you, but the computer will recognise them as three different variables.

Please re-read the last page before you read this one. Seriously, it will really help.

* * *

We can also use a variable as a subscript.

Subscripted variable: **P(K)**
Subscript: ↗

Call it "P sub K"

Below is the 1970 population and growth table.

* * *

REGIONS	K	P(K) *[in millions]*	R(K) %
AFRICA	1	344	2.6
ASIA	2	2056	2.3
N. AMERICA	3	228	1.1
L. AMERICA	4	283	2.9
EUROPE	5	462	0.8
U.S.S.R.	6	243	1.0
OCEANIA	7	19	2.0
WORLD	8	3635	2.0

* * *

P(K) is the population in *millions of people* for region K.
R(K) is the rate of growth expressed as per cent for region K.

For example, North America is region 3.

P(3) = 228 million people

R(3) = 1.1%

Your turn. Complete the following:

P(2) = __________ million people
R(2) = __________ %

P(8) = __________ million people
R(8) = __________ %

Since we are dealing with a new idea, we will apply it to an old problem. *[The logic of that escapes me, but it seems pedagogically sound. –Ed.]*

We write a program to read values of P(K) and R(K) and print the 1970 WORLD POPULATION DATA TABLE.

```
100 REMARK***READ THE P(K)'S AND R(K)'S
110 READ M
120 DIM P(M),R(M)
130 FOR K=1 TO M
140 READ P(K),R(K)
150 NEXT K
```

IF YOU GET AN ERROR MESSAGE ABOUT LINE 120, DELETE LINE 120 AND TRY AGAIN. WE'LL EXPLAIN LATER.

```
200 REMARK***PRINT THE HEADING
210 PRINT "1970 WORLD POPULATION DATA"
220 PRINT
230 PRINT "REGION NO.","POPULATION","GROWTH RATE"
240 PRINT

300 REMARK***PRINT THE TABLE
310 FOR K=1 TO M
320 PRINT K,P(K),R(K)
330 NEXT K

900 REMARK***DATA BASE
905 DATA 8
910 DATA 344,2.6,2056,2.3
920 DATA 228,1.1,283,2.9
930 DATA 462,.8,243,1
940 DATA 19,2,3635,2

999 END

RUN

1970 WORLD POPULATION DATA

REGION NO.     POPULATION     GROWTH RATE

 1              344            2.6
 2              2056           2.3
 3              228            1.1
 4              283            2.9
 5              462            .8
 6              243            1
 7              19             2
 8              3635           2

READY.
```

To find out how the program works, turn the page.

How does the program work?

1 Line 110 reads the value of M. Now the computer knows how many values of P(K) and R(K) are involved. Line 120 is a **DIM**ension statement which says *"Reserve M places in the computer memory for P(K)'s and M places for R(K)'s."*

BEWARE **Some BASIC systems do not permit a variable to appear in a DIM statement. If you have trouble, ask someone to explain how the DIM statement works on your computer or dig the information out of the reference manual for your system.**

---

2 Lines 130 to 150 cause the computer to **READ** the values of P(1), R(1), P(2), R(2), etc., into the computer's memory, so that they end up being stored like this:

P(1)	344	R(1)	2.6
P(2)	2056	R(2)	2.3
P(3)	228	R(3)	1.1
P(4)	283	R(4)	2.9
P(5)	462	R(5)	.8
P(6)	243	R(6)	1
P(7)	19	R(7)	2
P(8)	3635	R(8)	2

3 Lines 210 to 240 direct the computer to print the heading.

4 Lines 310 to 330 tell the computer to print M rows of numbers:

each row contains

the value of K
the value of P(K)
and the value of R(K).

Voilà!

BUILDING BLOCKS

Our programs are getting longer. Time to introduce another new idea . . .

Subroutines!

. . . featuring two new BASIC statements.

GOSUB & RETURN

The following program has a **MAIN PROGRAM**, three **SUBROUTINES**, and a data base. The subroutines are *called* by GOSUB statements in the main program. More about that later. Here is the program.

> *We will use the subroutines and data base again. Punch them on paper tape !!!*

```
100 REMARK***MAIN PROGRAM
110 READ M
120 DIM P(M),R(M)
130 GOSUB 310
140 GOSUB 410
150 GOSUB 510
160 STOP

300 REMARK***SUBROUTINE: READ P(K)'S AND R(K)'S
310 FOR K=1 TO M
320 READ P(K),R(K)
330 NEXT K
340 RETURN

400 REMARK***SUBROUTINE: PRINT HEADING
410 PRINT "1970 WORLD POPULATION DATA"
420 PRINT
430 PRINT "REGION NO.","POPULATION","GROWTH RATE"
440 PRINT
450 RETURN

500 REMARK***SUBROUTINE: PRINT THE TABLE
510 FOR K=1 TO M
520 PRINT K,P(K),R(K)
530 NEXT K
540 RETURN

900 REMARK***DATA BASE
905 DATA 8
910 DATA 344,2.6,2056,2.3,228,1.1,283,2.9
920 DATA 462,.8,243,1,19,2,3635,2

999 END
```

RUN the above program. The results should be the same as the results on Page 51.

The main program puts the building blocks together.

THE STATEMENT	TELLS THE COMPUTER
110 READ M	*READ the value of M. It is in the DATA statement, Line 905.*
120 DIM P(M), R(M)	*Allocate space in the computer's memory for P(1) through P(M) and for R(1) through R(M).*
130 GOSUB 310	*Do a subroutine, beginning at Line 310. On reaching a RETURN statement, return here, then move on to the next line.*
140 GOSUB 410	*Do a subroutine, beginning at Line 410. On reaching a RETURN statement, return here, then move on to the next line.*
150 GOSUB 510	*Do a subroutine, beginning at Line 510. On reaching a RETURN statement, return here, then move on to the next line.*
160 STOP	*Stop the computer.*

* * *

Try this . . . replace the main program, as follows:

```
100 REMARK***MAIN PR0GRAM
110 READ M
120 DIM P(M), R(M)
130 G0SUB 410 <--------------- Print the heading first.
140 G0SUB 310 <--------------- Then read the P(K)'s and R(K)'s.
150 G0SUB 510 <--------------- And then print the table.
```

RUN IT— Same Old Results...

WE HOPE YOU WILL GET INTO THE HABIT OF WRITING ALL YOUR PROGRAMS AS A MAIN PROGRAM THAT CALLS SUBROUTINES AS THEY ARE NEEDED. FROM NOW ON, WE WILL.

INFORMATION RETRIEVAL

Are the building blocks (*subroutines, that is)* and data base in the computer? If not, enter them (*from paper tape, we hope*). Then enter the following main program. It uses only *one* of the subroutines.

```
100 REMARK***MAIN PROGRAM
110 READ M
120 DIM P(M),R(M)
130 GOSUB 310
140 PRINT "INFO FOR WHICH REGION";
150 INPUT K
160 PRINT
170 PRINT "POPULATION IS";P(K);"MILLION"
180 PRINT "GROWTH RATE IS";R(K);"%"
190 PRINT
200 GO TO 140
```

```
RUN

INFO FOR WHICH REGION?1
```

Get the info for Asia.

```
POPULATION IS 344 MILLION
GROWTH RATE IS 2.6 %

INFO FOR WHICH REGION?5
```

Get the info for Europe

```
POPULATION IS 462 MILLION
GROWTH RATE IS .8 %

INFO FOR WHICH REGION? (and so on...)
```

But wouldn't it be nice if we could do it like this:

```
RUN

INFO FOR WHICH REGION? ASIA

POPULATION IS 344 MILLION
GROWTH RATE IS 2.6 %
```

(Perhaps you can! Ask someone (or check the reference manual) about **STRINGS** and **STRING VARIABLES** and **STRING FUNCTIONS.**
***** WARNING ***** once you start using string operations, you won't be able to do without them!!! They're addicting.)

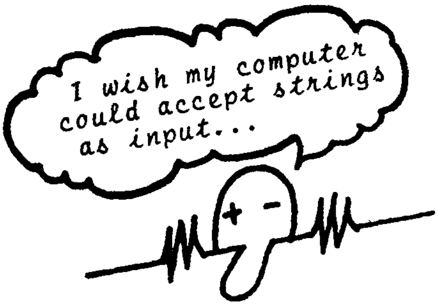
New main program (same old subroutines and data base).

```
100 REMARK***COMPARE TWO REGIONS, MAIN PROGRAM
105 READ M
110 DIM P(M),R(M)
115 GOSUB 310
120 GOSUB 410
125 GOSUB 510
130 PRINT
135 PRINT "LET'S COMPARE THE GROWTH OF TWO REGIONS, A AND B."
140 PRINT "WHEN I ASK, YOU ENTER THE REGION NUMBER FOR REGION A"
145 PRINT "AND THE NUMBER FOR REGION B AND THE YEAR FOR WHICH"
150 PRINT "YOU WANT A COMPARISON. I'LL DO THE REST."
155 PRINT
160 PRINT "REGION A";
165 INPUT A
170 PRINT "REGION B";
175 INPUT B
180 PRINT "YEAR";
185 INPUT Y
190 LET N=Y-1970
195 LET Q1=P(A)*(1+R(A)/100)↑N
200 LET Q2=P(B)*(1+R(B)/100)↑N
205 PRINT "REGION";A;":",INT(Q1+.5);"MILLION"
210 PRINT "REGION";B;":",INT(Q2+.5);"MILLION"
215 GO TO 155
```

Enter the above main program. Also enter the three subroutines and the data base. Then RUN the complete program. It should look like this:

```
1970 WORLD POPULATION DATA

REGION NO.      POPULATION     GROWTH RATE

 1               344            2.6
 2               2056           2.3
 3               228            1.1
 4               283            2.9
 5               462            .8
 6               243            1
 7               19             2
 8               3635           2

LET'S COMPARE THE GROWTH OF TWO REGIONS. A AND B.
WHEN I ASK, YOU ENTER THE REGION NUMBER FOR REGION A
AND THE NUMBER FOR REGION B AND THE YEAR FOR WHICH
YOU WANT A COMPARISON. I'LL DO THE REST.

REGION A?1
REGION B?4
YEAR?1984
REGION 1 :       493 MILLION
REGION 4 :       422 MILLION

REGION A?1
REGION B?4
YEAR?2001
REGION 1 :       762 MILLION
REGION 4 :       687 MILLION

REGION A?
```

← *Your turn. Carry on!*



DOUBLE SUBSCRIPTS

Our table of population and growth rate for eight population regions is a set of demographic statistics. Here is another way to store it in the computer.

D is a table (matrix, array) of demographic data.

D is arranged in rows and columns, like this

REGION	POPULATION		GROWTH RATE	
AFRICA	D(1,1)	344	D(1,2)	2.6
ASIA	D(2,1)	2056	D(2,2)	2.3
N. AMERICA	D(3,1)	228	D(3,2)	1.1
L. AMERICA	D(4,1)	283	D(4,2)	2.9
EUROPE	D(5,1)	462	D(5,2)	0.8
U.S.S.R.	D(6,1)	243	D(6,2)	1.0
OCEANIA	D(7,1)	19	D(7,2)	2.0
WORLD	D(8,1)	3635	D(8,2)	2.0

That's right. D(1,1) is a name of a box, location, place to store a number in the computer. So is D(1,2) and D(2,1) and D(5,2) and D(6,1) and D(7,2) and . . . . Complete the following.

- What number is in D(4,1)? ________
- What number is in D(8,2)? ________
- The population of N. America is in D(____, ____)
- The growth rate of U.S.S.R. is in D(____, ____)

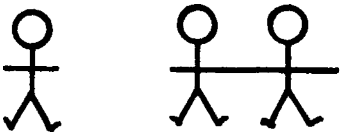
D has *two* subscripts: **D(3,2)**

left subscript *right subscript*

(first subscript) *(second subscript)*

In our example on the preceding page . . . the first subscript refers to the region.

It may be 1, 2, 3, 4, 5, 6, 7, 8.



The second subscript refers to the *kind* of data, 1 for population and 2 for growth rate.

We can also use variable subscripts.

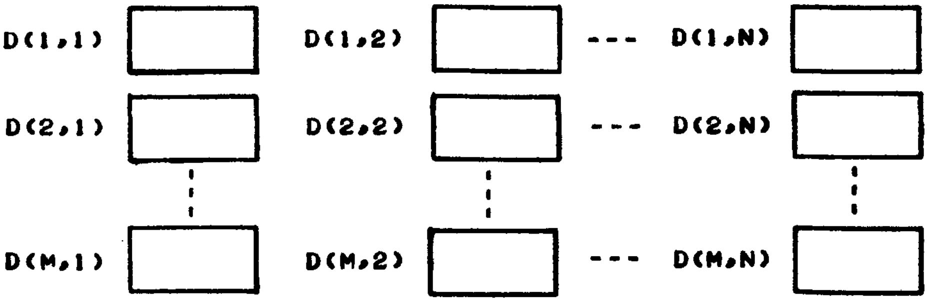
$$D(\underset{\uparrow}{K},\underset{\uparrow}{L})$$

Region (1–8)

L = 1 for population
L = 2 for growth rate

Our data base includes two kinds of data for 8 regions, (including the World total). We got our data from the 1970 WORLD POPULATION DATA SHEET. It lists data for 142 countries plus regional summaries. It also lists additional statistics for each country or region . . . for example, birth rate, death rate, and so on.

(IN GENERAL, THINK OF A DATA BASE FOR *M* CATEGORIES (CITIES, COUNTRIES, STATES, REGIONS, PLANETS, GALAXIES, CORPORATIONS, ...) WITH *N* STATISTICS FOR EACH CATEGORY.)



Here we go again . . . 1970 WORLD POPULATION DATA. This program is very similar to the program on Page 53. In fact, if you have that program on paper tape, load it and then make the changes indicated by the arrows.

```
100 REMARK***MAIN PROGRAM
110 READ M
120 DIM D(M,2) <---------This is a change.
130 GOSUB 310
140 GOSUB 410
150 GOSUB 510
160 STOP

300 REMARK***SUBROUTINE: READ DATA INTO D <----And this.
310 FOR K=1 TO M
320 READ D(K,1),D(K,2) <--------------Here is a change
330 NEXT K
340 RETURN

400 REMARK***SUBROUTINE: PRINT HEADING
410 PRINT "1970 WORLD POPULATION DATA"
420 PRINT
430 PRINT "REGION NO.","POPULATION","GROWTH RATE"
440 PRINT
450 RETURN

500 REMARK***SUBROUTINE: PRINT THE TABLE
510 FOR K=1 TO M
520 PRINT K,D(K,1),D(K,2) <------------And this is a change.
530 NEXT K
540 RETURN

900 REMARK***DATA BASE
905 DATA 8
910 DATA 344,2.6,2056,2.3,228,1.1,283,2.9
920 DATA 462,.8,243,1,19,2,3635,2


999 END

RUN

1970 WORLD POPULATION DATA

REGION NO.    POPULATION    GROWTH RATE

 1             344           2.6
 2             2056          2.3
 3             228           1.1
 4             283           2.9
 5             462           .8
 6             243           1
 7             19            2
 8             3635          2

READY.
```

things to do

- Replace the data base used in the program on Page 59 with the data base on Page 45. Then RUN the program.

- Rewrite the program on Page 56. Use the subscripted variable *D* instead of *P* and *R*. If you have everything on paper tape, it's easy!

- Here is a new data base.

REGION	POPULATION	GROWTH RATE	BIRTH RATE†	DEATH RATE†
AFRICA	344	2.6	47	20
ASIA	2056	2.3	38	15
N. AMERICA	228	1.1	18	9
L. AMERICA	283	2.9	38	9
EUROPE	462	.8	18	10
U.S.S.R.	243	1.0	17.9	7.7
OCEANIA	19	2.0	25	10
WORLD	3635	2.0	34	14

Rewrite the program on Page 59 to read the above data and print the table. Then think up ways to use the data and write programs to do so.

Suppose a data base has *M* countries with *N* statistics for each country? How would you write the program?

- Write a program to do this.

```
1970 WORLD POPULATION DATA

REGION NO.     POPULATION     GROWTH RATE

 1              344            2.6
 2              2056           2.3
 3              228            1.1
 4              283            2.9
 5              462            .8
 6              243            1
 7              19             2
 8              3635           2

DOUBLING TIME FOR WHICH REGION? 1
DOUBLING TIME IS ABOUT 27 YEARS

DOUBLING TIME FOR WHICH REGION?  ... AND SO ON
```

---

† *Births per 1000 population and deaths per 1000 population.*

Janus

Janus is a god. He has two faces. He looks backwards ... forwards.

Let's look backward, look forward.

- *Now you can speak a little BASIC.*
- *But you aren't yet fluent.*
- *We have introduced only a primitive form of BASIC. We did so because you may be using a computer that does not have the extended form of BASIC that permits use of*
    - *STRING variables and operations*
    - *MAT operations*
    - *FILES*
- *We have applied the computer to only one area of application ... population growth and demographic data.*

This is the end of the beginning. Look ahead ... and one more thing ...

please recycle this book

If you want to learn more about BASIC and you like math, try one of these:

- **Basic BASIC** by James S. Coan

  From: Hayden Book Company, Inc.
  116 West Fourteenth Street
  New York, NY 10011

- **BASIC PROGRAMMING** (Second Edition) by John G. Kemeny and Thomas E. Kurtz

  From: John Wiley and Sons, Inc.
  605 Third Avenue
  New York, NY 10016

If you want to learn more about BASIC and your math is a little wobbly (or non-existent!), try this one instead:

- **BASIC** by Robert L. Albrecht, LeRoy Finkel and Jerald R. Brown

  From: John Wiley and Sons, Inc.
  605 Third Avenue
  New York, NY 10016

If you want to learn more about computers, what they are, what they do, etc., the best book is:

- **Computers and Computation** by Scientific American

  From: W.H. Freeman and Company
  660 Market Street
  San Francisco, California 94104

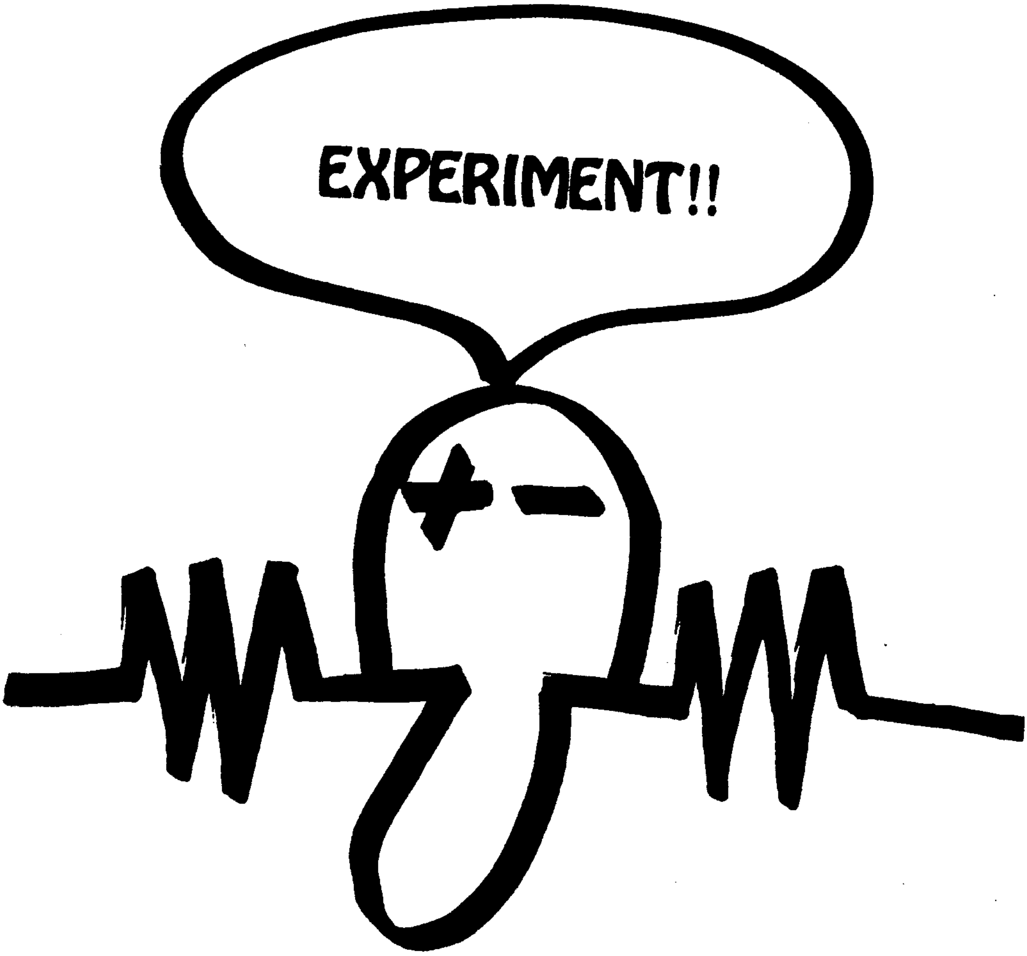
If you are having trouble thinking up things to try on your computer, here is a beautiful book . . . a classic:

- **Problems for Computer Solution** by Fred Gruenberger and George Jaffray

  From: John Wiley and Sons, Inc.
  605 Third Avenue
  New York, NY 10016

* * *

And maybe you would like to subscribe to PEOPLE'S COMPUTER COMPANY the newspaper about educational/recreational use of computers. See page 64 for details.


EXPERIMENT!!
+ –

dymax

p.o. box 310 menlo park, ca 94025

GAMES, TRICKS AND PUZZLES
FOR A HAND CALCULATOR, Wallace Judd
Dymax, 1974, 100pp., paperback

Do you own a hand calculator? You do? Good! This informative, entertaining, and very useful volume was written just for you.

The author has covered many of your favorite mathematical games and recreations. He shows you how a calculator can be used to perform many of the more common mathematical operations, such as how to extract square roots and percents, how to get the power of a number, and even how to generate random numbers. You are also shown how to perform tricks unique to the calculator and its keyboard. In addition to all of this, a typical calculator's insides are exposed to view and explained so that you can learn how it was put together and how it functions, and thus more readily understnad the hints given by the author on how to detect and correct some of the more common malfunctions hand calculators are prone to. The author ends his book by answering the questions asked in the text, of which there are many, and by giving the solutions to all of the problems and puzzles he posed to the reader.

$2.95 plus $0.50 handling to –

DYMAX
P. O. Box 310
Menlo Park, California 94025

---

is a newspaper about . . .

having fun with computers
learning how to use computers
how to buy mini-computers
books, films, music
tools of the future

PEOPLE'S COMPUTER COMPANY
is published 6 times during the year.

Single subscriptions $5 for 6 issues
[$6 outside U.S.A.]

Send check or money order to

PEOPLE'S COMPUTER COMPANY
P. O. Box 310
Menlo Park, California 94025

WHAT TO DO AFTER YOU HIT RETURN
OR
PCC's FIRST BOOK OF COMPUTER GAMES
158 pages, 1975, paperback

This book is destined to become one of those books -- It is conspicuous, one of those books that is too big to fit on the shelf, so you find it lying about on a table; it is eclectic -- one of those new, soft-cover, newsprint catalogs that is crammed to the margins with interesting tidbits and graphics; it is a curiosity -- one of those books you feel compelled to pick up, just to see what is inside; and most important, it is an educational resource one of those books that will help you find, obtain, or "get into" new materials for the enrichment of learning.

$6.95 plus $0.50 handling to --

PEOPLE'S COMPUTER COMPANY
P. O. Box 310
Menlo Park, California 94025

Dymax, P.O. Box 310, Menlo Park, California 94025