AUTHOR Hoitsma, Louis
TITLE BASIC in Ten Minutes a Day.
INSTITUTION Dartmouth Coll., Hanover, N.H. Kiewit Computation Center.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE Feb 70
GRANT NSF-GW-2246
NOTE 33p.; For related documents, see SE 028 536-538; Not available in hard copy due to copyright restrictions

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Computer Oriented Programs; Computers; Learning Activities; Mathematics Curriculum; *Mathematics Education; *Mathematics Instruction; *Mathematics Materials; *Programing; Secondary Education; *Secondary School Mathematics

ABSTRACT

This teacher's guide is designed for teaching BASIC and introducing students to the use of the computer by means of a series of short written lessons. The purpose of such a series is so that classroom time required for the series would not seriously hinder the progress of the normal course of study, and each student would have an opportunity to acquire substantial hands-on experience at the teletype while completing assignments during the six- or seven-week period planned to complete the unit. Twelve worksheets are given along with brief comments for teachers. (MP)

BASIC IN TEN MINUTES A DAY

by

Louis Hoitsma
Phillips Academy

SECONDARY SCHOOL PUBLICATION

KIEWIT COMPUTATION CENTER
DARTMOUTH COLLEGE
HANOVER, NEW HAMPSHIRE 03755

(Supported in part by National Science Foundation Grant No. GW-2246)

February 1970

TABLE OF CONTENTS

eries would not seriously hinder the progress of the normal ourse of study, and (2) each student would have an oppor-unity to acquire substantial 'hands on' experience at the eletype while completing assignments during the six or seven eek period planned to complete the unit. Once completed, he future is open to subsequent course-related applications nd individual exploration by the students.

he first lesson requires about fifteen minutes of class ime and about twenty minutes in the teletype room. Several essons in the series are self-explanatory, others involve xplanation or question periods which generally can be imited to ten minutes. Assignments can be scheduled for ompletion in two, three or four days, depending upon the egree of difficulty and the normal work load.

n undertaking this series, the teacher should make a specific ffort not to provide too much information (or all the answers) ut allow the students to do some thinking, searching and uestioning on their own; consequently, the question sessions ill often be lively. No particular stress was placed upon fficiency in writing programs, but this remained a challenge o a portion of the students. No part of the series was required', nor was pressure applied to complete assignments.

n a first-year algebra course, this project could be under-aken in the winter, soon after the students have some signifi-ant algebra under their belts, or in the spring. Higher-level ath courses might benefit from starting it earlier in the chool year. This approach might also prove to be practical or instructing 'uncomputerized' math teachers.

SHEET 1

Perhaps this is your initial exposure to a computer program. Think of a computer program as a set of directions issued in numerical order and, for the moment, imagine yourself as the computer confronted with this set of directions. What responses can you visualize making to each of the following programs?

PRINT-1

```
10 READ X
20 PRINT X
30 DATA 19
99 END
```

PRINT-2

```
10 READ A,B,C
20 PRINT A,B,C
30 DATA 19,3.2,-35
99 END
```

PRINT-3

```
10 READ X
20 PRINT X    (or 20 PRINT X,)
              (or 20 PRINT X;)
30 GO TO 10
40 DATA 19,-3.2,-7,0
99 END
```

COLUMNS

```
10 PRINT "FIRST NUMBER:", "SECOND NUMBER"
20 PRINT
30 READ X, Y
40 PRINT X, Y
50 GO TO 30
60 DATA 4,54,6,8,-4.5,1,348,0,-16,7
70 END
```

SUM-1

```
10 READ A,B
20 PRINT A+B
30 DATA 7,56
40 END
```

*NOTE: The two "or" statements are alternative to line 20 and should be so indicated by a bracket.

SUM-2

```
10 READ A,B

20 READ C

30 LET S = A+B+C

40 PRINT S

90 DATA 8,45,-3

99 END
```

BIGSUM-1

```
10 LET S=0

20 READ A

30 LET S = S+A

40 PRINT A,S

50 GO TO 20

90 DATA 7,56,-34,17

91 DATA 16,1,6.6,-3

92 DATA 13,3,5,43,0,5

99 END
```

BIGSUM-2

```
20 READ A

30 IF A=0 THEN 60

40 LET S = S+A

50 GO TO 20

60 PRINT S

90 DATA 7,56,-34,17,16,1

91 DATA5.6,-3,13,3,5,43,0,5

99 END
```

Now a few comments about these programs. Note that each line has a number and a line instruction. The line instructions you see on privious page (READ,PRINT, GOTO, LET, IF ...THEN, DATA END) are standard terminology in the language that we use to communicate with the computer at Dartmouth. The language is called BASIC. Two punctuation marks (, and ;) appear in the abo programs. Two additional self-explanatory instructions, LIST and RUN, put the computer to work.

A 15-minute explanation in the computer room should enable you to activate the teletype and start on your first assignment. Before doing your assignment you may first wish to type-in one of the above programs to get a LIST and RUN. Don't hesitate to ask questions if you get stuck.

INFORMATION COVERED DURING 15-MINUTE SESSION AT TELETYPE

Hand out a brief statement on activating procedure.
Go through the process of activating the TTY -- ORIG, FDX,
    User number, OLD, NEW, HELLO
Panel buttons -- ORIG, CLR, FDX, K, T, KT (on Model 35 ASR)
Keyboard
    Only upper case letters
    Special keys -- carriage return, line feed
    Distinguish between 1 and L, 0 and Ø
Have a student at the TTY or type a simple program
READY, LIST, RUN
Deletions: back-arrow, Control-X
Operation symbols: *,/, +, -
Other symbols: -,
Symbols not on the TTY: ≠ - , , , . ,
Variables; single letter or single letter followed by a single
    digit.

General procedures for programming:

Write your program before sitting at the TTY.
Try your program manually several times before TTY time.

BASIC Manual }
BASIC Programming text } Both available in the room.
Library programs ***
BASICT*** - a self-teach program
SAVE, UNSAVE, Storage
Sign-up routine
Care of the teletype
Condition <u>of</u> the room
Conditions <u>in</u> the room

FIRST ASSIGNMENT

1. Write a program (call it BEGIN-1) that will have the computer print your name.

2. Write a program (BEGIN-2) that will compute and print the product of two numbers.

3. Write a program (BEGIN-3) that will read successive pairs of numbers and, on each pass, print out the numbers and their product. Enter at least 12 numbers in DATA.

Submit a LIST and RUN on one sheet.

SHEET #2

The following program lists the odd integers from 1 to 25 and their squares.

PRINTSQ

```
20 PRINT "X", "THE SQUARE ØF X"        or  { 19 PRINT " ","2"
                                           { 20 PRINT "X","X "
30 PRINT

40 FOR X = 1 TØ 25 STEP 2

50 PRINT X,X↑2  (NOTE: X↑2 means X²)

60 NEXT X

70 END
```

SECOND ASSIGNMENT

Name of program: 3-TABLE

Problem: To construct a table of squares and cubes of the multiples of 3 from 12 to 42, inclusive. You are to produce a printed table with headings, giving the <u>multiples</u>, their <u>squares</u> and their <u>cubes</u>.

Submit a LIST and RUN.

```
PRINT X,
PRINT X;
```

Do you recall the RUN of each?

PRINT-3 introduced a simple loop routine (lines 10-30), using a GØ TØ statement. CØLUMNS did the same in lines 30-50.

SUM-2 used a LET statement with a variable (line 15) to represent the sum, A+B.

The two BIGSUM programs got you more involved in programming, perhaps just because they contained more lines. Both of these programs have a GØ TØ loop (lines 20-50), and the latter has an exit from the loop in the forms of an IF...THEN statement (line 25). This exit eliminates the OUT OF DATA statement observed in the RUN of BIGSUM-1.

In the FIRST assignment several of you had a RUN of BEGIN-3 which printed the product of the two numbers on a separate line from the numbers. What does it take to keep the product written to the right of the two numbers on the same line? Could you alter BEGIN-3 so that the RUN would read:

```
     2 X 3 = 6                        THE PRØDUCT ØF 2 AND 3 IS 6
                       or
     9 x 7 = 63                       THE PRØDUCT ØF 9 AND 7 IS 63
```

For assistance and reminders on later assignments it might be worth saving these sheets and your programs. Also, a BASIC Manual is located in the Computer Room for your use; a personal copy can be obtained at the bookstore.

Sheet #2 introduced you to a second looping routine, FØR and NEXT (lines 40-60). This is a slick way of increasing the values of a variable in a systematic manner. Here is another example of a FØR and NEXT loop:

LØØP

```
10 FOR X=2 TØ -1 STEP -.5
20 PRINT X
30 NEXT X
40 END
```

The RUN will look like this:

```
  2
  1.5
  1
 .5
-.5
-1
```

In a FØR and NEXT loop, the statement <u>FØR X = B TØ E STEP P</u> is written with B denoting the beginning of the sequence, E the end of the sequence, and P the step size through the sequence. B, E and P can be decimal or integer, positive or negative; but the step P must be in the direction of B toward E, of course. It goes without saying that P shouldn't exceed the difference between B and E.

Look over this program to calculate the sum of all positive integers greater than zero up to a given integer N.

<u>LØØP-1</u>

```
10 READ N
20 LET D = 1
30 LET S = 0
35
40 LET S = S+D
50 IF D = N THEN 80
60 LET D = D + 1
70 GØ TØ 40
75
80 PRINT S
90 DATA 10
99 END
```

D in line 20 is a counting variable. NOTE that it takes on successive values of 1, 2, 3, ... as the program recycles through the loop. Can you tell that the loop runs from line 40 - 70? Line 50 checks during each cycle to see if it is time to leave the loop.

THIRD ASSIGNMENT (From BASICT Sequence)

The objective of this assignment is to write programs using three different types of loops: (1) GØ TØ, (2) IF....THEN, and (3) FØR and NEXT.

1. Using LOOP-1 as a pattern, make appropriate changes in it to write a program to calculate the product of all positive

integers <u>less than</u> a given integer N----call the program LØØP-2.

2. Try to rewrite LØØP-2 without using the GØ TØ statement----name it LØØP-3.

   HINT - change your IF...THEN statement.

   NOTE - the line number following the THEN part of the statement may be an earlier line in the program

3. Write a program LØØP-4 that computes the sum of a sequence of numbers. Use the FØR and NEXT looping technique and READ B,E and P from a DATA statement. (See the lesson)

SUBMIT a LIST and RUN

SHEET #4

Suppose N, D and Q are positive integers. If N>D then

$$\frac{N}{D} = Q + \frac{\text{Remainder}}{D}$$

From this equation only two different situations can arise, one with remainder $\neq 0$, the other with remainder = 0.

Situation A

If N = 11 and D = 3,

$$\frac{11}{3} = 3 + \frac{(\text{remainder of } 2)}{3} = 3 + \frac{2}{3}.$$

Written as a decimal,

$$3 + \frac{2}{3} = 3.6666\underline{6} = 3.66667.$$

If one (1) is subtracted from 3.66667 once, again, and again, eventually the difference will be a negative number, but never zero. Are you able to tell why?

Situation B

If N = 20 and D = 5,

$$\frac{20}{5} = 4 + \frac{(\text{Remainder of } 0)}{5} = 4.$$

Subtract 1 several times and you reach zero before your difference is a negative number. In this situation there is something special about the relationship between N and D.

Situation B tells us that N is divisible by D, which is not the case in situation A.

FOURTH ASSIGNMENT

Your fourth assignment is to write a program to determine the divisibility of two positive integers, N and D, with N D, by carrying out successive subtractions of one from the quotient N/D, as presented above.

Name your program DIVIS.

Suggested procedure: After introductin two variables and reading in a value for each, you will need a LET statement to produce a quotient, then a loop to subtract one from your quotient a sufficient number of times to obtain either a zero or negative result. When this occurs you should set up an exit from the loop for each case, providing directions to proceed to an appropriate PRINT statement.

Submit a LIST and RUN.

Note to teacher: It is suggested that a handout of the following three programs not be presented to the students until they complete the fourth assignment

```
DIVIS     13:12     04/29/68
10 READ N,D
20 LET Q=N/D
30 LET R=Q
40 LET R=R-1
50 IF R=0 THEN 80
60 IF R<0 THEN 90
70 GØ TØ 40
80 PRINT N"IS DIVISIBLE BY "D" AND THE QUØTIENT IS "Q
85 STØP
90 PRINT N"IS NØT DIVISIBLE BY"D
100 DATA 18,2
110 END


LIST

DVIS-INT     08:35     04/30/68

10 INPUT N,D
20 LET Q = N/D
30 IF Q = INT(Q) THEN 80
40 IF Q<>INT(Q) THEN 90
80 PRINT N "IS DIVISIBLE BY "D" AND THE QUØTIENT IS "Q
85 GØ TØ 10
90 PRINT N "IS NØT DIVISIBLE BY "D
95 GØ TØ 10
100 END

READY

RUN

DVIS-INT     08:35     04/30/68

? 18,2
 18 IS DIVISIBLE BY 2 AND THE QUØTIENT IS 9
? 34,7
 34 IS NØT DIVISIBLE BY 7
? 12,3
 12 IS DIVISIBLE BY 3 AND THE QUØTIENT IS 4
? STØP
PROGRAM HALTED

TIME:     .06 SEC.
```

14

SHEET #5

The last program you wrote, DIVIS, probably resembled the one reproduced by the teacher.

Introduction of the line instruction INPUT and the function INT, signifying 'the greatest integer contained in', allows us to be a bit more sophisticated in our program writing. The program DVIS-INT, also reproduced on the previous page, enables you to follow the development.

Information to note regarding the

1. INPUT statement

   a) The INPUT statement provides for the entry of data during the RUN of the program. The INPUT statement acts as a READ statement but does <u>not</u> draw numbers from a DATA statement.

   b) In DVIS-INT, the user types two numbers, separated by a comma, presses the return key and the computer goes on with the rest of the program.

   c) There might be a PRINT statement combined with the INPUT statement to make sure that the user knows what the question mark is asking for. You could type:

   ```
   20 PRINT "YØUR VALUES ØF X,Y AND Z ARE";
   30 INPUT X,Y,Z
   ```

   and in the RUN the computer will type out:

   ```
   YØUR VALUES ØF X,Y AND Z ARE ?
   ```

   Without the semicolon at the end of line 20, the question mark would have been printed on the next line.

2. INT function

   a) The INT function outputs the greatest integer not greater than x. Thus INT(12.3) = 12, INT(-2.1) = -3, INT(3) = 3.

FIFTH ASSIGNMENT

1. Write a program, FACTØRS, that will determine <u>all</u> divisors (factors) of a positive integer and print them.

   Special instructions (or hints)

   a) INPUT your number, say N.

b) You will need a loop to provide integers, D, to divide into N; preferably a FØR and NEXT loop.

c) As for the loop, would 1 be your first value of D, or could it be 2? How many integers between 1 and N would have to be divided into N to get all possible divisors?

d) Using the INT function, if your quotient, Q is equal to the greatest integer no greater than Q, then you have a "divisor". If Q $\neq$ INT(Q), ignore that value of D and go on to the next.

2. This program takes off in a different direction ----- call it WAGES. Write it to compute the net wages of employees of the KIEWIT Company. You are given the gross wages of the employees below, and the deductions are: income tax of 20% if income is $80 or less, income tax of 22% if income is more than $80, union dues 1% of gross wages, and FICA (or Social Security) 4 and 1/2% of the <u>first</u> $90 of earnings per week. The employees wages are $200, $162, $148, $67, $72, $100, $90, $80, $106, $150, $50, $10. Label column headings: GRØSS WAGES, TAXES, UNIØN, FICA, NET PAY.

Submit a LIST and RUN.

16

SHEET #6

Can you produce the RUN of the program CØUNT-1 in pencil?

```
CØUNT-1                  10 LET T = 0
                         20 READ A
                         30  PRINT T;A
                         40 LET T = T + 1
                         50 GØ TØ 20
                         60 DATA 13,9,20,43,74,121
```

This isn't exactly what we want for a counting procedure. What change would you make in the program in order to have the number 6 and 121 matched up on the bottom line of the RUN?

Now for a second program involving counting:

```
CØUNT-2                  10 LET N = 1
                         20 LET P = 1
                         30 READ A
                         40 IF A = 0 THEN 80
                         50 LET P = P*A
                         60 LET N = N+1
                         70 GØ TØ 30
                         80 PRINT "THE PRODUCT OF THE "N" NUMBERS IN
                            DATA IS "P"."
                         90 DATA 34,12,2,6,4,7,84,34,56
                         91 DATA 0
                         99 END
```

CØUNT-2 has an incorrect number of entries stated in the RUN. Take a look!

```
RUN
COUNT-2
THE PRODUCT OF THE 10 NUMBERS IN DATA
IS 2.19253 E+10.
```

Can you correct this 'error'?

SIXTH ASSIGNMENT

1. Write a program, CØUNT-AV, which will read a group of at least six numbers from DATA, count them, determine their average and printout the average, the number of entries and the entries.

   Example of printout:

```
5.5 IS THE AVERAGE OF THE 8 NUMBERS:

8, 2, 14, 5, 3, 7, 1, 4
```

If you write the program with separate loops for counting, for determining the sum, and for printout of entries, you must have a READ in each loop using the line instruction RESTØRE before each loop. Use of RESTØRE can be quickly understood by looking on page 52 of the beige BASIC manual in the computer room. It is not essential to use three loops, however, but it just might be simpler at the start.

SHEET #7

(Parts of the following are from materials developed by Mr Peyton Pitney of the Mount Hermon School.)

The purpose of this lesson is to investigate an iterative scheme for finding an approximate square root of a real number. We had several methods for finding approximations in class previously; but Newton's Method will enable you to sense the power and speed of the computer.

The basis of Newton's Method for approximating square roots is the observation that if you choose exactly the right answer for the square root of a number n, then the quotient of n and this choice is the choice itself. For example, if you choose 42 as the square root of 1764 and divide 1764 by 42, the quotient is 42. Notice that the divisor and the quotient are the same when your choice is correct. However, if you guess the wrong answer (for example 40.4) for the square root of 1764 and divide 1764 by 40.4, the quotient is not 40.4 but about 43.66. In other words, if your choice is incorrect, either the divisor or the quotient will be too small, and the other too large; furthermore, the correct answer will be between the divisor and the quotient.

In using Newton's Method we make a succession of guesses, each successive guess being the average of the previous guess (divisor) and the quotient.

Illustration: Find an approximate square root of 751.48.

```
First guess:  20
First quotient:  751.48/20 = 37.574
First average:   (37.574 + 20)/2 = 28.787

Second guess:  28.787
Second quotient:  751.48/27.787 = 26.105
Second average:   (28.787 + 26.105)/2 = 27.446

Third guess:  27.446
Third quotient:  751.48/27.446 = 27.380
Third average:   (27.446 + 27.380)/2 = 27.413

etc.
```

SEVENTH ASSIGNMENT: Write a program, NEWTØNSQ, which will enable any user to find an approximation of the square root of any positive number. The RUN should provide an answer in the form, "THE APPRØXIMATE PØSITIVE SQUARE- RØØT ØF ... IS ...."

Submit a LIST and RUN.

NOTE: One important line in your program involves a new function, ABS, signifying absolute value. You could use it as follows:

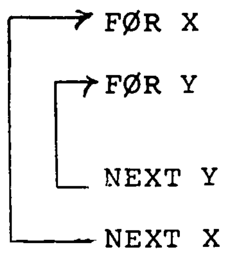
```
If ABS(N - D) < .000001 THEN ...
```

provides for additional iterations if the difference between divisor and quotient is not less than .000001.

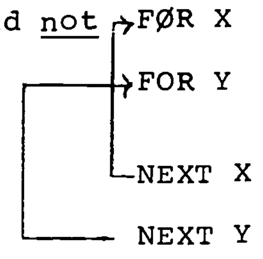
Extra for experts: After completing the above program compose one which will find the cube root of a number using a method analogous to NEWTONSQ.

SHEET #8

The next computer routine involves NESTED LOOPS, which are loops within loops, best expressed with FOR and NEXT statements. The loops must actually be nested, and must not cross. A skeleton example:



and not



The program, RØØTS, which follows, illustrates the use of the double-loop. RØØTS is designed to print a table of square, cube and fourth roots. The two loops are nested. The outside loop is for the integers, N, of which we are to take the roots. The inside one is needed since for each N we wish to computer three different roots. Here we use R in the sense of the Rth root; a square-root, a cube-root, and a fourth-root, all of which are written as the 1/R power of N.

```
RØØTS     15:01     05/20/68

10 PRINT "NUMBER", SQUARE RØØT", "CUBE RØØT", "FØURTH RØØT"
20 FØR N = 1 to 10
30    PRINT N,
40    FØR R = 2 TØ 4
50       PRINT N↑ (1/R),
60    NEXT R
70    PRINT
80 NEXT N
90 END
```

NUMBER	SQUARE RØØT	CUBE RØØT	FØURTH RØØT
1	1	1	1
2	1.41421	1.25992	1.18921
3	1.73205	1.44225	1.31607
4	2.	1.5874	1.41421
5	2.23607	1.70998	1.49535
6	2.44949	1.81712	1.56508
7	2.64575	1.91293	1.62658
8	2.82843	2.	1.68179
9	3.	2.08008	1.73205
10	3.16228	2.15443	1.77828

(After all your work with NEWTØNSQ it may be comforting to learn that the computer can provide these roots simply by asking for them. There is also a function, SQR, which approximates $\sqrt{62}$, for example, by writing LET X = SQR(62).

The next program to write is called PUZZLE which finds all three-digit numbers which are equal to the sum of the cubes of their digits.

Mathematics involved: Recall that when we write a numeral with three digits, each digit has a "place value". For example, "435 represents 4*100 + 3*10 +5. With this idea in mind you should understand that 100*h + 10*t + u is a general representation for a three-digit number, if h, t and u designate the hundreds', tens' and units' digits, respectively.

In approaching this problem in a methodical way one would assign a value to h, then assign a value to t, and "run through" ten different values for u, testing each time to see if the sum of their cubes equals the three-digit number. Afterwards, another value would be assigned to t, and one would check again the ten different values for u, and so on.

EIGHTH ASSIGNMENT:

1. Write PUZZLE, which finds all three-digit numbers which are equal to the sum of the cubes of their digits. Submit LIST and RUN.

   (Can you visualize the change(s) necessary in PUZZLE to enable it to find all integers less than 10,000 which are equal to the sum of their digits?)

2. A program not related to the above one is the solution of a quadratic equation. Write a program which will find the

solutions of a quadratic equation ---- call it QUADSØL. Write it so that you can INPUT the coefficients. Make allowances for equations with no real solution.

SHEET #9

(Parts of the following are from materials developed by Mr. Peyton Pitney of the Mount Hermon School.)

A routine that should be of enough value to you in future math or science programming is that of round-off. A few examples will assist.

a) Suppose that we wish to round off 71.176 to the nearest hundredth of a unit.

i)	Multiply by 100	7117.6
ii)	Add .5	7118.1
iii)	Determine the greatest integer which is less than or = this sum	7118
iv)	Divide by 100	71.18
v)	In general, INT(X*100+.5)/100	

b) Suppose that we wish to round off 71.176 to the nearest tenth of a unit.

i)	Multiply by 10	711.76
ii)	Add .5	712.26
iii)	Determine the greatest integer which is < or = this sum	712
iv)	Divide by 10	71.2
v)	In general, INT(X*10 + .5)/100	

c) A typical instruction in a given program might be

```
55 PRINT INT(N*1000 + .5)/1000
```

which would provide the value of N, rounded off to the nearest thousandth of a unit.

The program RNDØFF supplies the decimal equivalents to the rational numbers 1/7, 2/7, 3/7, ... 6/7; these equivalents are rounded off in the first column to 6 places (by the computer, without instruction), and in the remaining columns to 5, 4, 3, 2 and 1-decimal place, respectively, in the RUN.

```
RUNDØFF     08:38     05/24/68

10  FØR N = 1 TO 6
20  LET Q = N/7
30  LET Ø = INT(Q*10 + .5)/10
40  LET T = INT(Q*100 + .5)/100
50  LET H = INT(Q*1000 + .5)/1000
60  LET F = INT(Q*10000 + .5)/10000
70  LET V = INT(Q*100000+ .5)/100000
80  PRINT N"/7 ="Q;V;F;H;T;Ø
90  NEXT N
100 END
```

```
1 /7 = 0.142857  0.14286  0.1429  0.143  0.14  0.1
2 /7 = 0.285714  0.28571  0.2857  0.286  0.29  0.3
3 /7 = 0.428571  0.42857  0.4286  0.429  0.43  0.4
4 /7 = 0.571429  0.57143  0.5714  0.571  0.57  0.6
5 /7 = 0.714286  0.71429  0.7143  0.714  0.71  0.7
6 /7 = 0.857143  0.85713  0.8571  0.857  0.86  0.9
```

NINTH ASSIGNMENT

1. Write a program, name it USEFUL, that will generate the first ten positive integers and calculate their squares, cubes, square roots (rounded off to three-decimal places) and cube roots (rounded off to two-decimal places). Print in columns with headings.

2. Name this program SKETCH. Let $y = x^2 + 2x - 8$ in line 10. For values of x from -5 to 3 at intervals of .5, have your program direct the computer to print the values of x and the corresponding values of y under proper column headings. Then take the ordered pairs (x, y) from your RUN, plot them on a graph and join these points with a smooth curve.

   or (if functions have been covered at this point in the course)

   Using a new line instruction, DEF, define the second degree function F(X) in line 10 as follows: 10 DEF FNF(X)=X↑2+2*X-8 Have the computer evaluate F(X) at intervals of .5 on the domain $-5 \leq X \leq 3$. Then using graph paper and letting y = F(X) plot the ordered pairs (x,y) and join them with a smooth curve.

25

SHEET #10

Once you read a list of numbers it is often useful to attach different name or labels to each of them. Imagine that you want to read and label the prices of twelve different articles at the corner store. You could say:

```
        READ P0, P1, P1,P2,...., P8, P9, R0, R1
```

Each of these labels plays the role of a different variable and in BASIC these are legitimate variables.

However, the idea of a list is very similar to the notion of a subscripted variable. Subscripted variables are the topic of this sheet. A mathematician might write L_3 to indicate the value of the third number in the list L. In BASIC, it would be written L(3). For this reason we often call the numbers in a list, subscripted variables.

Assume you want to read nine numbers in list L.

```
10 FØR I = 1 TØ 9
20   READ X(I)
30 NEXT I
40 DATA 4,7,2,-5,1,4,6,20,70
50 END
```

Each number in DATA now has a name: X(1)=4,X(2)=7,X(3)=2,....X(6)=4, X(9)=70.

The computer will automatically reserve up to ten spaces for the numbers in a list, so if your list (or subscripts on a variable) contains more than ten numbers, you must indicate the desired dimensions explicitly in a DIM(dimension) statement in your program. Suppose the program had sixteen numbers in DATA, then

```
 5 DIM X(16)
10 FØR I = 1 TØ 16
20 etc., with 16 items in DATA
```

The loop has now named your sixteen numbers in the list.

Now let us read through pairs of numbers.

```
 5 DIM X(20), Y(20)
10 FØR I = 1 TØ 20
20   READ X(I), Y(I)
30 NEXT I
40 DATA 2,3,30....( with 40 numbers in all)
50 END
```

As a final example, suppose that the first number in the data stack tells how many triples (in this case) are to be read. No DIM statement is necessary with fewer than ten spaces to reserve for one variable. In the following program, TRIØ, observe line 50 ....do you see what it produces in the RUN?

```
LIST

TRIØ     18:24     06/27/68

30 READ N
40 FØR I = 1 TØ N
50   READ X(I), Y(I), Z(I)
60   PRINT "X("I")="X(I),"Y("I")="Y(I),"Z("I")="Z(I)
70 NEXT I
80
90 DATA 4
95 DATA 3,12,4,5,-9,0,7,2,-1,-3,7,6
99 END

READY

RUN

TRIØ     18:24     06/27/68

X( 1 ) = 3      Y( 1 ) = 12     Z( 1 ) = 4
X( 2 ) = 5      Y( 2 ) = -9     Z( 2 ) = 0
X( 3 ) = 7      Y( 3 ) =  2     Z( 3 ) = 1
X( 4 ) =-3      Y( 4 ) =  7     Z( 4 ) = 6

TIME:     .07 SECS.
```

TENTH ASSIGNMENT

1. Write a program which reads and prints two lists, X and Y, the first with 12 numbers and the latter containing 15 number.

2. Refer to Problem 1. Set up a list S that consists of the sums of the first twelve corresponding numbers of list X and list Y (that is, $S_i = X_i + Y_i$ for i = 1 to 12). Call the program LISTSUM.

Extra for experts: Five articles in a store are priced $4, $1.50, $2.20, $.70, and $1.10, respectively. Assign subscripts P(1), P(2),.....P(5) to these prices. Three customers, A, B and C make purchases of these items as follows:

	PRICE				
	$4	$1.50	$2.20	$.70	$1.10
No. bought by A	5	2	1	3	5
No. bought by B	1	0	5	1	3
No. Bought by C	3	5	2	7	1

Write a program using subscripted variables A(I), B(I), and C(I), along with P(I) to determine the bill for each customer separately and then the total bill.

SHEET #11

We want to write a program to rearrange a list of numbers in order of decreasing magnitude. We can approach this routine by moving through the list comparing L(1) and L(2), interchanging the numbers in the pair if their order is not what we want, then comparing the number in position L(2) with L(3), interchanging if their order is not correct, then moving to the numbers in position L(3) and L(4), and so on through the list.

Suppose we have four numbers 12, 25, -1, and 38 to rearrange into decreasing order.

On our first pass through the list,
compare 12 with 25---interchange them---the new order is 25,12,-1,38;
compare 12 with -1---no change---order remains the same
compare -1 with 38---interchange them---order now is 25,12,38,-1.

We now pass through the list a second time.
We first compare 25 with 12---no change---order is 25,12,38,-1;
12 compared with 38---interchange them---new order is 25,38,12,-1;
12 compared with -1---no change---order remains the same.

And another pass through the list.
25 compared with 38---interchanged---order is now 38,25,12,-1;
and comparisons of 25 with 12 and 12 with -1---no changes, so the final order is 38,25,12,-1.

The process takes some concentration and a bit of 'hand computing' to get it throughly under control. The program ØRDER is an example to work on.

Lines 20-40 assign subscripts to each number in the list T. Lines 60-140 go through the ordering routine, rearranging in order of decreasing value. To accomplish this trick a double loop is used and adjacent pairs of numbers are compared. In line 100, if $T(1) \geq T(2)$ then proceed to compare T(2) with T(3). But if $T(1)<T(2)$, lines 110 and 120 interchange T(1) and T(2), exchanging both names and position in the list. The new T(2) is now compared with T(3), and so on. Comparing T(2) with T(3) comes on the second pass through the inner loop. T(3) and T(4) are compared on the third pass through the inner loop. When T(1) and T(2) are again compared we are in the second pass through the outer loop. Lines 160-180 print the list in the final order.

Note that ORDER requires no DIM statement!

```
ORDER      12:53      06/28/68


20 FØR I = 1 TØ 8
30   READ T(1)
40 NEXT I
50
60 FØR I = 1 TØ 8
70   FØR J = 1 TØ (8-I)
80        LET X = T(J)
90        LET Y = T(J+1)
100       IF X > = Y THEN 130
110       LET T(J+1) = X
120       LET T(J) = Y
130  NEXT J
140 NEXT I
150
160 FØR I = 1 TØ 8
170   PRINT T(I);
180 NEXT I
190
900 DATA 12,45,-14,65,23,-6,78,38
999 END


RUN


ØRDER     12:54     06/28/68


78  65  45  36  23  12  -6  -14


TIME:      .07 SECS.
```

ELEVENTH ASSIGNMENT

1. Eleven students earn grades of 95, 63, 68, 70, 84, 98, 78, 75, 90, 57, 78.
   Copying these grades into a DATA statement in the given order, write a program to arrange and print the grades in increasing order. Name the program LØ-HI, and SAVE it.

2. Rename (type:RENAME) the program in problem 1, calling it HI-LØ, and alter it to print the same grades in decreasing order. SAVE it.

3. Rename the program in problem 2, call it HIGHEST, and adjust it to print only the highest grade in the group.

4. Can you use or adjust one of the above programs to print only the honor grades in the given list, in decreasing order? (An honor, grade, for consideration here is 80 or better.) Call this program HØNØRS.

SHEET #12

One additional statement, REMARK or REM, is used to provide information for you and/or anyone else reading your program, to describe it or give directions for using the program; it provides no information for the computer. In fact, the computer ignores REMARKS (saying, in effect, "He is only talking to himself, not to me.") To indicate a remark, REMARK or REM is typed following the line number of any line in the program.

Example:

```
10 REM....THIS PRØGRAM PLØTS THE GRAPH ØF A
          FUNCTIØN.

15 REM

20 REM DEFINE YØUR FUNCTIØN IN LINE 100 AND

30 REM ENTER THE RIGHT AND LEFT ENDPØINTS(etc.)...
```

The sheets and assignments you have worked with in this series have given you an adequate start in understanding and using BASIC and for programming on the GE-635 computer at Dartmouth. Numerous other routines and capabilities can be learned from the BASIC manual, from other users, and by doing more programming on your own. Now you ought to try out your own ideas in some programs.

Some additional thoughts might come from such a list as this:

1. Evaluate and find zeros of a function.
2. Find integral solutions of an equation in two unknowns. (Diophantine Equations)
3. Find all the factors of a number.
4. List all prime numbers less than a given number. (Sieve of Eratosthenes)
5. Determine the prime factors of a number.
6. Find the GCD and LCM of two numbers.
7. Read about Fibbonacci numbers, then write a program to produce them.
8. Convert linear measures from one system to another, such as meters and centimeters to feet and inches.
9. Solve a system of simultaneous equations (2 variables, 3 variables, more?)
10. Produce a list of integers which satisfy the Pythagorean Theorem.

COMMENTS FOR TEACHERS

SHEET 1 Go right to wrok getting class reactions after handing out the sheet to each student. Responses and explanations should be limited to 15 minutes. It is helpful to show a RUN of each program (including each alternative for line 20 in the program PRINT-3), either by individual handout or acetate on overhead projector. Anticipate a host of questions on the 2nd day -- but try to limit it to ten minutes.

SHEET 2 It might be helpful to have a RUN of PRINTSQ and the alternate to show in class. The TAB instruction could be introduced at this point.

SHEET 3 It is worthwhile to go over the summary with the class when this sheet is distributed, having the RUN of programs on Sheet 1 to refer to. Save the new material on this sheet 1 until the next day. On problem #1 it is presumed that students will follow the pattern of LØØP-1 with a GØTØ loop, otherwise problem #2 might not make sense.

SHEET 4 This is motivation for INT. Some students will need help with the notion presented on this sheet, particularly with regard to handling Q as it decreases. This seemed to be a worthy lesson.

SHEET 5 Hand out the RUN of DIVIS and DVIS-INT with this sheet. Problem #1 part (c) ---- the answer to the second question, N/2, is a point for mathematical discussion. Problem #2 demands care in organization. The FICA deduction is based on the first $90 of everyone's pay regardless of how much he earns ---- that point may not be clear.

SHEET 6 A few minutes of discussion on CØUNT-1 is helpful, but let then wrestle with CØUNT-2 overnight. CØUNT-2 offers an opportunity to point out the meaning of "2.19253 E+10". Suggest playing 'computer' by hand. Assistance with the developemnt of the assigned program is suggested, for it seems simpler to understand, organize the write using three separate loops and the instruction RESTØRE. CØUNT-1 can be corrected by changing line 10 to read: 10 LET T=1

SHEET 7 The illustrated example ought to be explained and another example (perhaps on acetate) displayed, for only then will all the reading make sense to most.

ABS (N-D)   .000001 might need explanation as to its role.

SHEET 8 Has the meaning of "1/R", as an exponent, been covered? Point out SQR(62) on bottom of page, students do not like to read an explanation such as the one on the top of the second page in this lesson, but they ought to be asked to explain it on the second day -- for they should know what is being talked about at this point in the course. If the solution of quadratics by formula has not been covered as yet, a substitute for problem #2 in the assignment could be to write a program to produce the prime numbers up to 100.

SHEET 9 Note Problem #2 ------ which alternative suits best?

SHEET 10 Subscripts could be used elsewhere in first-year algebra in the handling of 'proportions' and 'slope of a line'.

SHEET 11 This sheet ought to be given a thorough going-over with the class. Once they get through the routine by hand a few times, the notion seems to strike home.