Requirements:
A floppy diskette is required for storing programs.

The following programs should be typed in as an exercise in seeing how Flow Control programs work, and the use of their programs. At the end of the programs to be typed in are exercises that you should complete to help you better understand the programming instructions listed below.

Program 2.4.1: - Flow Control entry-condition

```
REM
'
CONST MaxPower = 1000
Power = 1   ' The first power n^0 is always 1

COLOR 10, 1
CLS
PRINT "This program prints all powers < 1000 of an Inte-
ger."
INPUT "Enter an Integer"; n

DO WHILE Power < MaxPower
  PRINT Power;
  Power = Power * n
LOOP
```

Deskcheck 2.4.1:

Output 2.4.1:

Program 2.4.2: - Flow Control entry-condition

```
'
'
‘DO [{WHILE | UNTIL} condition]
‘    [statementblock]
‘LOOP
‘
' Declare variables to use and default values
DIM goal AS DOUBLE
DIM interest AS DOUBLE
DIM payment AS DOUBLE
DIM years AS INTEGER
DIM balance AS DOUBLE
balance = 0

' Get some data from the user
CLS
COLOR 14,1
PRINT "So you want to put some money aside for retirement."
INPUT "How much money do you need to retire"; goal
INPUT "How much money will you contribute every year"; pay-
ment
INPUT "Interest rate in % (eg. use 7.5 for 7.5%)"; interest

' Let's make some calculations
interest = interest / 100
DO WHILE balance < goal
   balance = (balance + payment) * (1 + interest)
   years = years + 1
LOOP

PRINT
PRINT "You can retire in "; years; " years"
PRINT "with "; balance; "in the bank"
```

Deskcheck 2.4.2:

Output 2.4.2:

Programming Exercises

1. Modify the program 2.4.1 to print both the power and its value: For example:

```
Integer: 2
Power    Value
0        1
1        2
2        4
3        8
```

2. Rewrite the 2.3.3 program using a DO LOOP entry-condition loop instead of the for loop. The program adds all the numbers between 1 and *n* (given by the program user) and displays the result.

3. Draw the flow-charts for both 2.3.3 and Question 1 and make a comparison statement highlighting differences between the two loop structures related to the problem solved.

4. Draw the flow-chart for Program 2.4.1

5. Rewrite Program 2.4.1 as an exit-condition loop.

6. Modify program 2.4.1 so that after a table is written on the screen, the program asks the user whether they want to see another table. If the user wants to see another table, the program goes back to the beginning of the program and starts again.