ALGORITHMS AND FLOWCHARTS

ALGORITHMS AND FLOWCHARTS

- A typical programming task can be divided into two phases:
- ***Problem solving phase***
  - produce an ordered sequence of steps that describe solution of problem
  - this sequence of steps is called an ***algorithm***
- ***Implementation phase***
  - implement the program in some programming language

Steps in Problem Solving

- First produce a general algorithm (one can use ***pseudocode***)
- Refine the algorithm successively to get step by step detailed ***algorithm*** that is very close to a computer language.
- ***Pseudocode*** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.

Pseudocode & Algorithm

- **Example 1:** Write an algorithm to determine a student’s final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Pseudocode & Algorithm

Pseudocode:

- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

  Print "FAIL"

  else

  Print "PASS"

Pseudocode & Algorithm

- Detailed Algorithm
- Step 1: Input M1,M2,M3,M4

  Step 2: GRADE $\leftarrow$ (M1+M2+M3+M4)/4

  Step 3: if (GRADE < 50) then

  Print "FAIL"

  else

  Print "PASS"

  endif

The Flowchart

- (Dictionary) A schematic representation of a sequence of operations, as in a manufacturing process or computer program.
- (Technical) A graphical representation of the sequence of operations in an information system or program. Information system flowcharts show how data flows from source documents through the computer to final distribution to users. Program flowcharts show the sequence of instructions in a single program or subroutine. Different symbols are used to draw each type of flowchart.

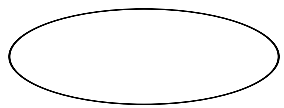
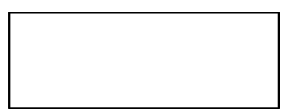
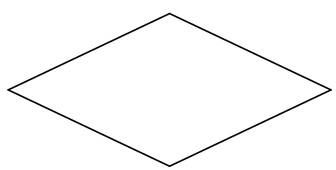
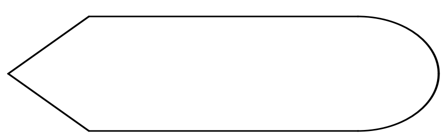
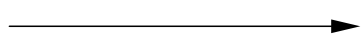
The Flowchart

A Flowchart

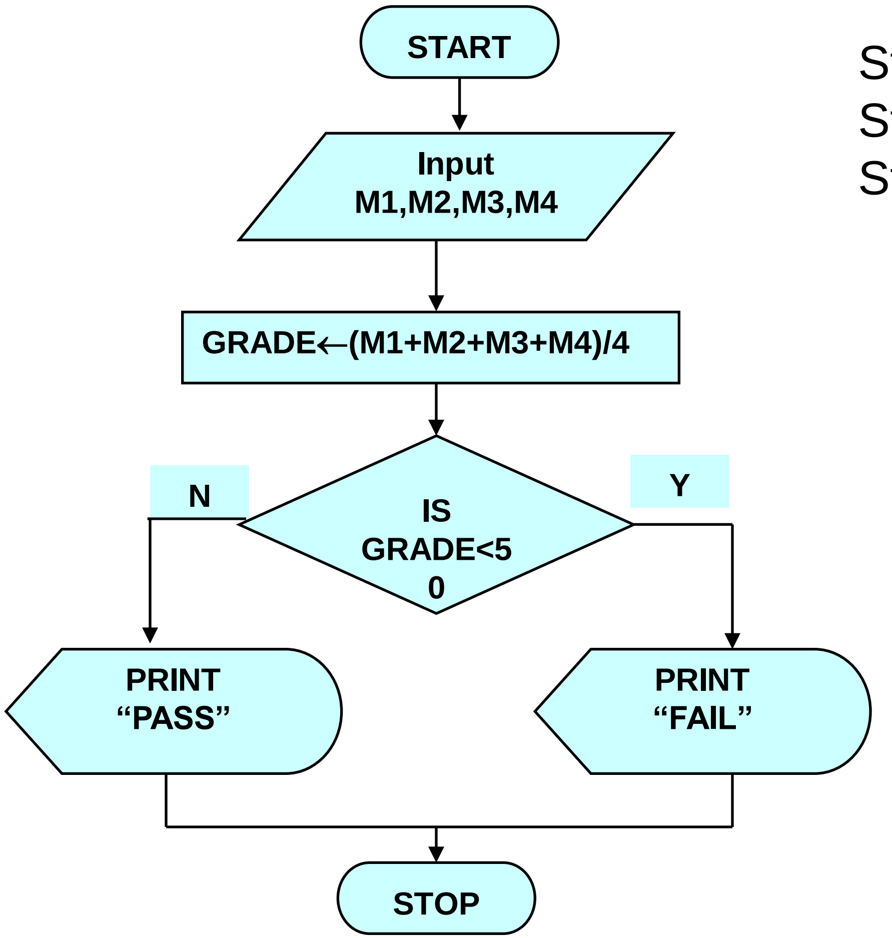
- shows logic of an algorithm
- emphasizes individual steps and their interconnections
- e.g. control flow from one action to the next

Flowchart Symbols

Basic

Name	Symbol	Use in Flowchart
Oval		Denotes the beginning or end of the program
Parallelogram		Denotes an input operation
Rectangle		Denotes a process to be carried out e.g. addition, subtraction, division etc.
Diamond		Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)
Hybrid		Denotes an output operation
Flow line		Denotes the direction of logic flow in the program

Example



Step 1: Input M1,M2,M3,M4
Step 2: GRADE ← (M1+M2+M3+M4)/4
Step 3: if (GRADE <50) then
Print “FAIL”
else
Print “PASS”
endif

Example 2

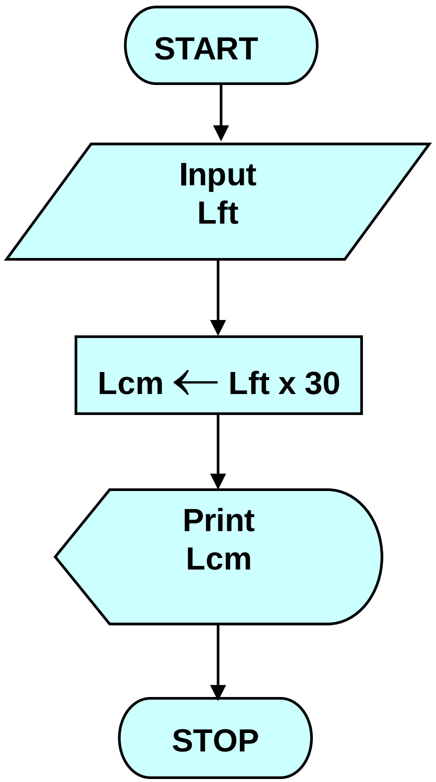
- Write an algorithm and draw a flowchart to convert the length in feet to centimeter.

Pseudocode:

- *Input the length in feet (Lft)*
- *Calculate the length in cm (Lcm) by multiplying LFT with 30*
- *Print length in cm (LCM)*

Example 2

Flowchart



Algorithm

- Step 1: Input Lft
- Step 2: Lcm ← Lft x 30
- Step 3: Print Lcm

Example 3

Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

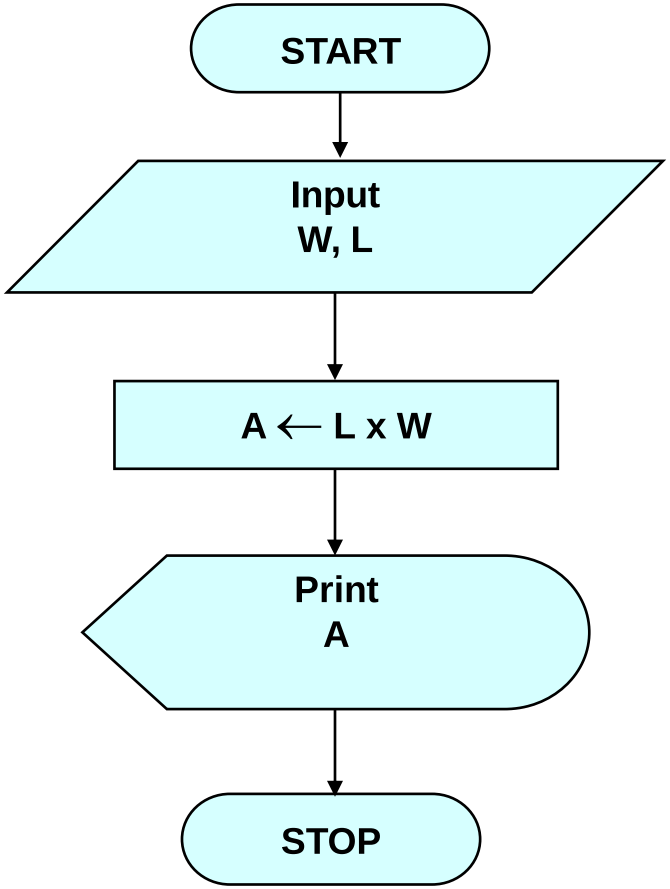
Pseudocode

- *Input the width (W) and Length (L) of a rectangle*
- *Calculate the area (A) by multiplying L with W*
- *Print A*

Example 3

Algorithm

- Step 1: Input W,L
- Step 2: $A \leftarrow L \times W$
- Step 3: Print A



Example 4

- Write an algorithm and draw a flowchart that will calculate the roots of a quadratic equation $ax^2 + bx + c = 0$
- Hint: **d** = sqrt ($b^2 - 4ac$), and the roots are: $\boldsymbol{x}\mathbf{1} = (-b + d)/2a$ and $\boldsymbol{x}\mathbf{2} = (-b - d)/2a$

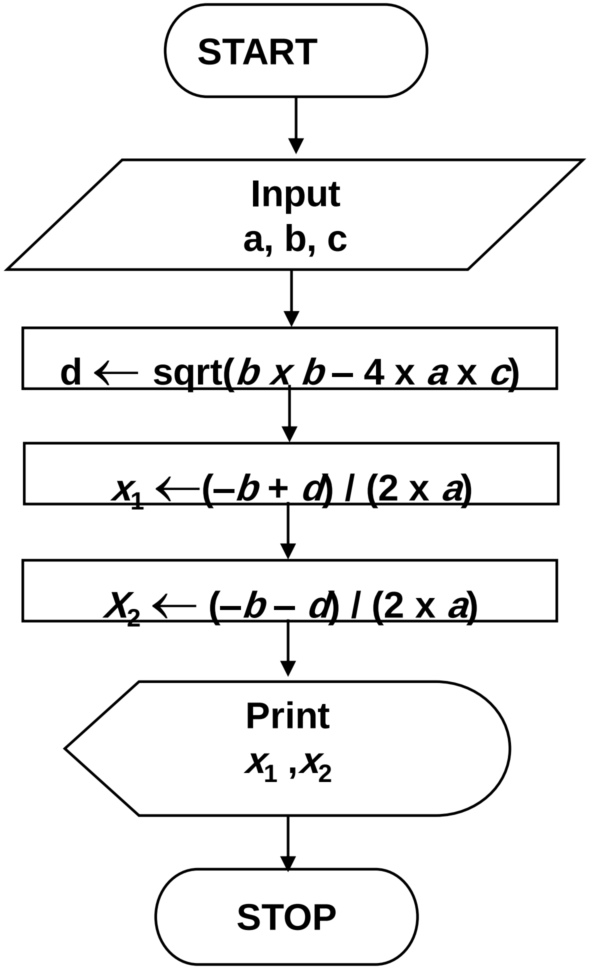
Example 4

Pseudocode:

- *Input the coefficients (a, b, c) of the quadratic equation*
- *Calculate* **d**
- *Calculate* ***x*1**
- *Calculate* **x2**
- *Print* x*1 and* x2

Example 4

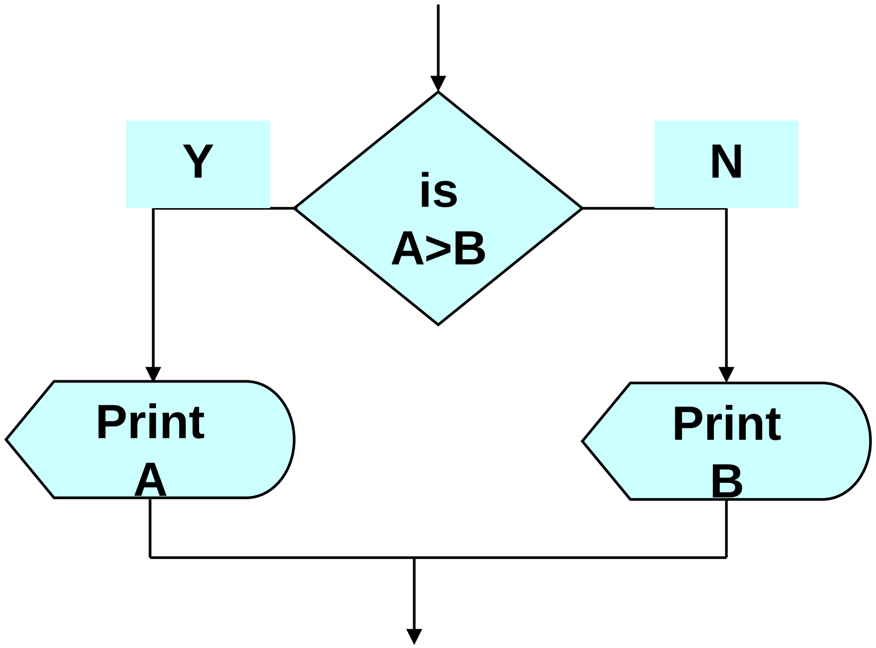
- **Algorithm**:
- Step 1: Input a, b, c
- Step 2: $d \leftarrow$ sqrt ($b \times b - 4 \times a \times c$)
- Step 3: $x1 \leftarrow (-b + d) / (2 \times a)$
- Step 4: $x2 \leftarrow (-b - d) / (2 \times a)$
- Step 5: Print $x1$, $x2$



DECISION STRUCTURES

- The expression A>B is a logical expression
- *it describes a **condition** we want to test*
- ***if A>B is true (if A is greater than B)** we take the action on left*
- print the value of A
- ***if A>B is false (if A is not greater than B)** we take the action on right*
- print the value of B

DECISION STRUCTURES



IF–THEN–ELSE STRUCTURE

- The structure is as follows

If condition then

    true alternative

  else

    false alternative

endif

IF–THEN–ELSE STRUCTURE

- The algorithm for the flowchart is as follows:

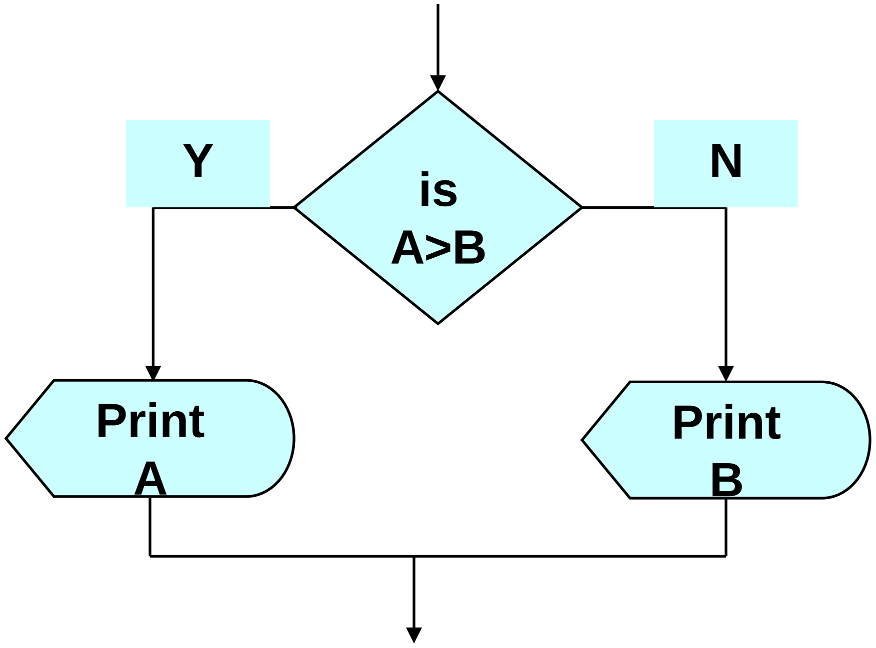
If A>B then

print A

else

print B

endif



Relational Operators

Relational Operators	
Operator	**Description**
>	Greater than
<	Less than
=	Equal to
≥	Greater than or equal to
≤	Less than or equal to
≠	Not equal to

Example 6

- Write an algorithm that reads two values, determines the largest value and prints the largest value with an identifying message.

ALGORITHM

Step 1: *Input* VALUE1, VALUE2

Step 2: *if* (VALUE1 > VALUE2) *then*

MAX $\leftarrow$ VALUE1

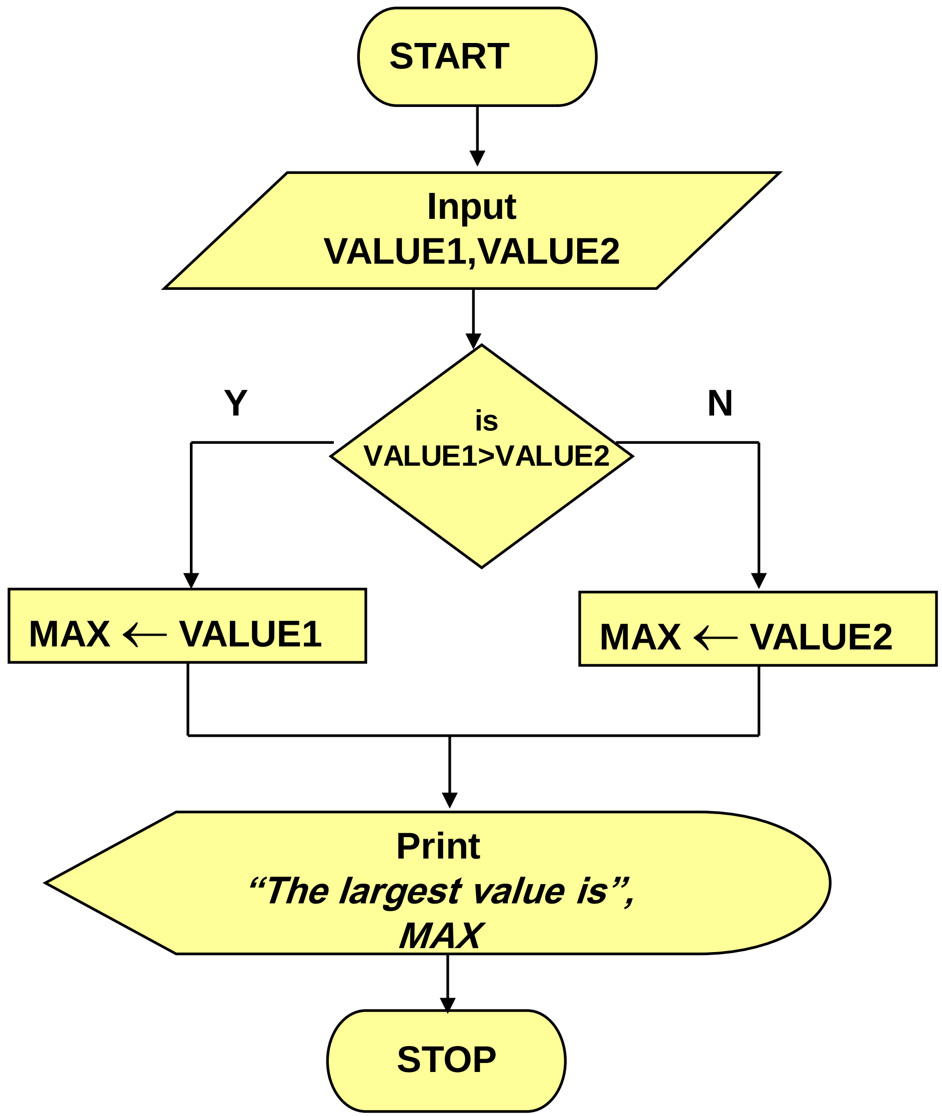
else

MAX $\leftarrow$ VALUE2

endif

Step 3: *Print "The largest value is", MAX*

Example 6



START
Input
VALUE1,VALUE2
is
VALUE1>VALUE2
Y
N
MAX ← VALUE1
MAX ← VALUE2
Print
"The largest value is",
MAX
STOP

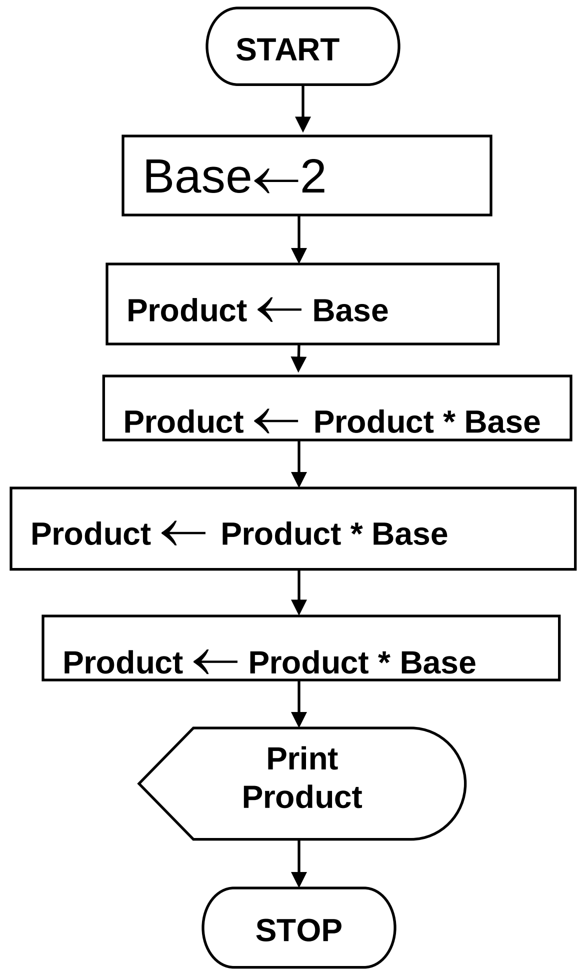
LOOPS

- Computers are particularly well suited to applications in which operations are repeated many times.
- If the same task is repeated over and over again a loop can be used to reduce program size and complexity

Example 7: Write an algorithm and draw a flowchart to calculate 2^4 .

- **Algorithm**:
- Step 1: Base $\leftarrow$ 2
- Step 2: Product $\leftarrow$ Base
- Step 3: Product $\leftarrow$ Product * Base
- Step 4: Product $\leftarrow$ Product * Base
- Step 5: Product $\leftarrow$ Product * Base
- Step 6: *Print* Product

Flowchart


START
Base←2
Product ← Base
Product ← Product * Base
Product ← Product * Base
Product ← Product * Base
Print Product
STOP

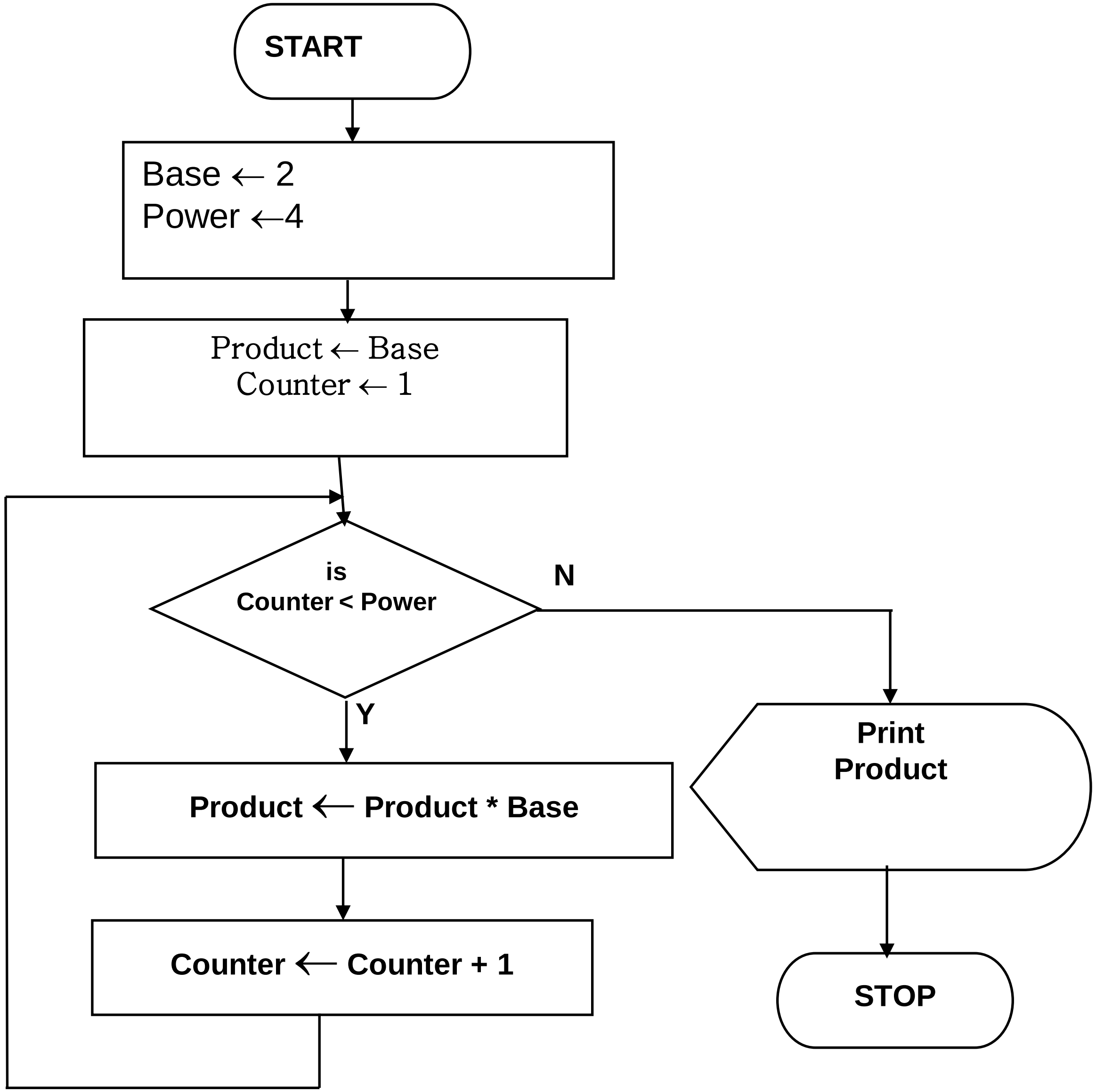
- Question: What happens if you want to calculate 2 to the power of 1000?
- **Answer**: Use a LOOP (repeated execution of the same set of instructions)

Example 8:

- Write an algorithm and draw a flowchart to calculate 2^4 using a loop approach? Verify your result by a *trace table*.

Algorithm:

Step 1: Base $\leftarrow$ 2
Step 2: Power $\leftarrow$ 4
Step 3: Product $\leftarrow$ Base
Step 4: Counter $\leftarrow$ 1
Step 5: While Counter < Power
Repeat Step 5 through step 7
Step 6: Product $\leftarrow$ Product * Base
Step 7: Counter $\leftarrow$ Counter +1
Step 8: *Print* Product


START
Base ← 2
Power ←4
Product ← Base
Counter ← 1
is
Counter < Power
N
Y
Product ← Product * Base
Counter ← Counter + 1
Print
Product
STOP

TRACING

	BASE	POWER	PRODUCT	COUNTER	COUNTER < POWER
STEP 1:	2	?	?	?	?
STEP 2:	2	4	?	?	?
STEP 3:	2	4	2	?	?
STEP 4:	2	4	2	1	T
STEP 5:	**2**	**4**	**2**	**1**	**T**
STEP 6:	2	4	2x2=4	1	T
STEP 7:	2	4	4	1+1=2	T
STEP 5:	**2**	**4**	**4**	**2**	**T**
STEP 6:	2	4	4x2=8	2	T
STEP 7:	2	4	8	2+1=3	T
STEP 5:	**2**	**4**	**8**	**3**	**T**
STEP 6:	2	4	8x2=16	3	T
STEP 7:	2	4	16	3+1=4	F
STEP 5:	**2**	**4**	**16**	**4**	**F**

STEP 8: print **16**.

Step 1: Base ← 2

Step 2: Power ← 4

Step 3: Product ← Base

Step 4: Counter ← 1

Step 5: While Counter < Power
Repeat Step 5 through step 7

Step 6: Product ← Product * Base

Step 7: Counter ← Counter +1

Step 8: *Print* Product

- **Example 10:** Write down an algorithm and draw a flowchart to find and print the largest of three numbers. Read numbers one by one. Verify your result by a trace table. (Use 5, 7, 3 as the numbers read)

Algorithm

- Step 1: *Input* N1
- Step 2: Max ← N1
- Step 3: *Input* N2
- Step 4: *If (*N2>Max) *then*

  Max = N2

  endif
- Step 5: *Input* N3
- Step 6: *If (*N3>Max) then

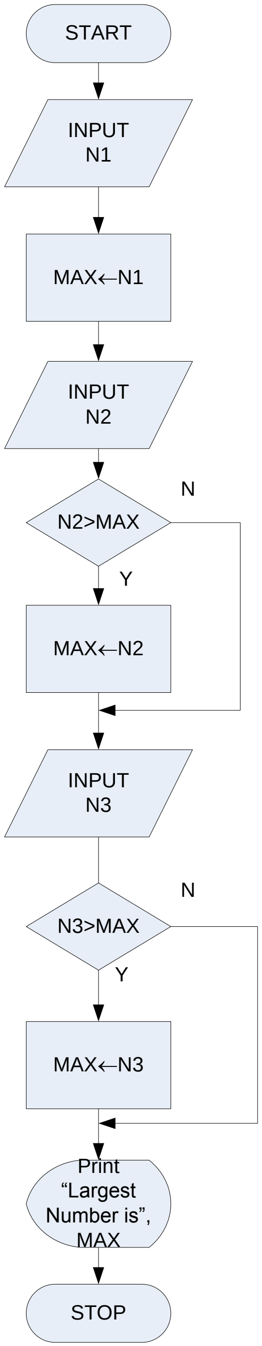
  Max = N3

  endif
- Step 7: *Print* "The largest number is:",Max

Flowchart & Tracing

	N1	N2	N3	Max	N2>Max	N3>Max
Step1:	5	?	?	?	?	?
Step 2:	5	?	?	5	?	?
Step 3:	5	7	?	5	T	?
Step 4:	5	7	?	7	T	?
Step 5:	5	7	3	7	F	F
Step 6:	5	7	3	7	F	F

Step 8: Print → Largest Number is **7**

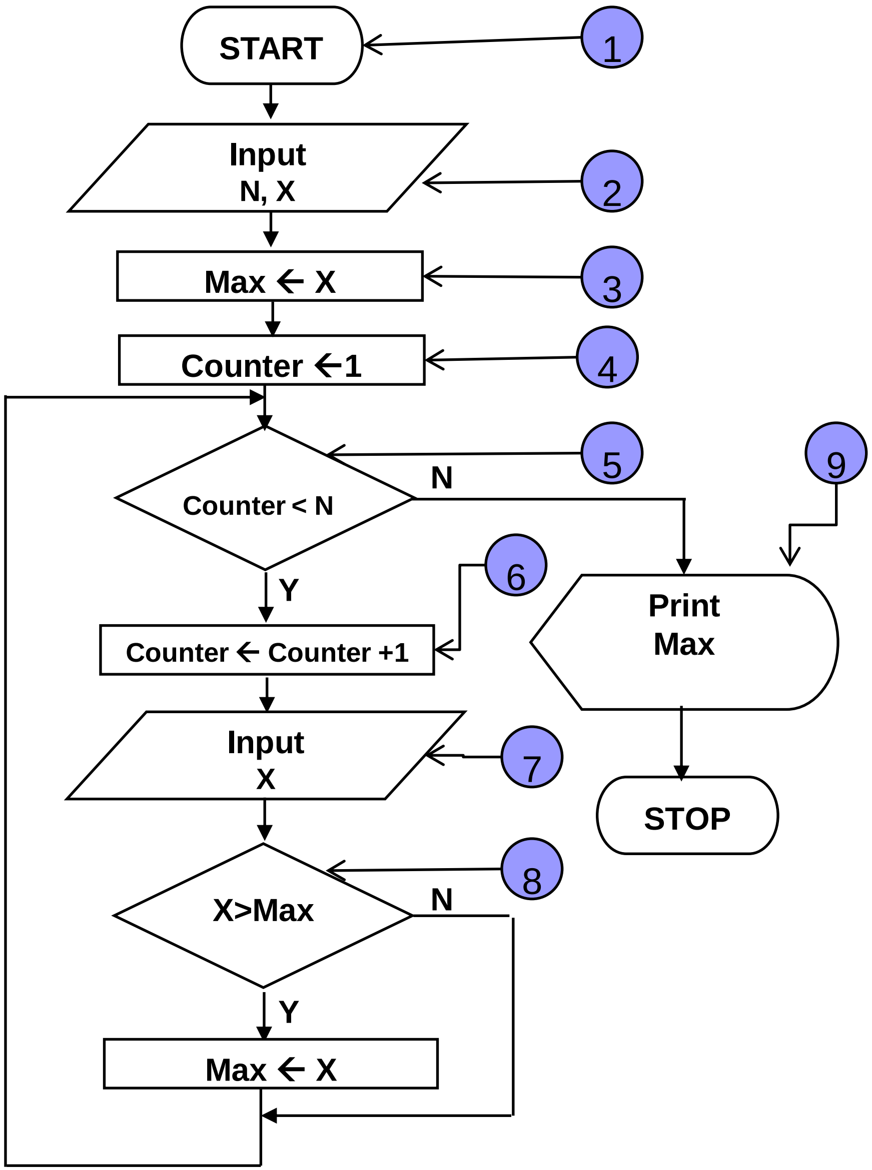


- **Example 11**: Write down an algorithm and draw a flowchart to find and print the largest of N (N can be any number) numbers. Read numbers one by one. Verify your result by a trace table. (Assume N to be 5 and the following set to be the numbers {1 4 2 6 8 })

Algorithm:

- Step 1: *Input* N
- Step 2: *Input* X
- Step 3: Max ← Current
- Step 4: Counter ←1
- Step 5: *While* (Counter < N)
  Repeat steps 5 through 8
- Step 6: Counter ← Counter + 1
- Step 7: *Input* X
- Step 8: *If* (X > Max) then
  Max ← X
  endif
- Step 9: *Print* Max

Tracing

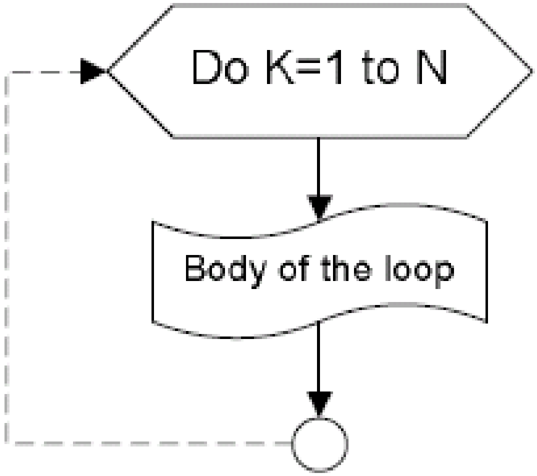


	N	X	Max	Counter	Counter < N	Next > Max
Step 1	5	1				
Step 2	5	1				
Step 3	5	1	1			
Step 4	5	1	1	1	T	
Step 5	5	1	1	1	T	
Step 6	5	1	1	2	T	
Step 7	5	4	1	2	T	
Step 8	5	4	4	2	T	T
Step 5	5	4	4	2	T	F
Step 6	5	4	4	3	T	F
Step 7	5	2	4	3	T	F
Step 8	5	2	4	3	T	F
Step 5	5	2	4	3	T	F
Step 6	5	2	4	4	T	F
Step 7	5	6	4	4	T	T
Step 8	5	6	6	4	T	T
Step 5	5	6	6	4	T	F
Step 6	5	6	6	5	F	F
Step 7	5	8	6	5	F	T
Step 8	5	8	8	5	F	T
Step 5	5	8	8	5	F	F
Step 9			**8 output**			

How many times will steps 4, 6, and 7 be executed?

Do Loops

- It is convenient to introduce a special type of loop that is headed by a special macroinstructions.
- This terminology comes from FORTRAN , although many programming languages have this type of loop.



- For example :

BASIC

DO K=1 to N

{body of loop}

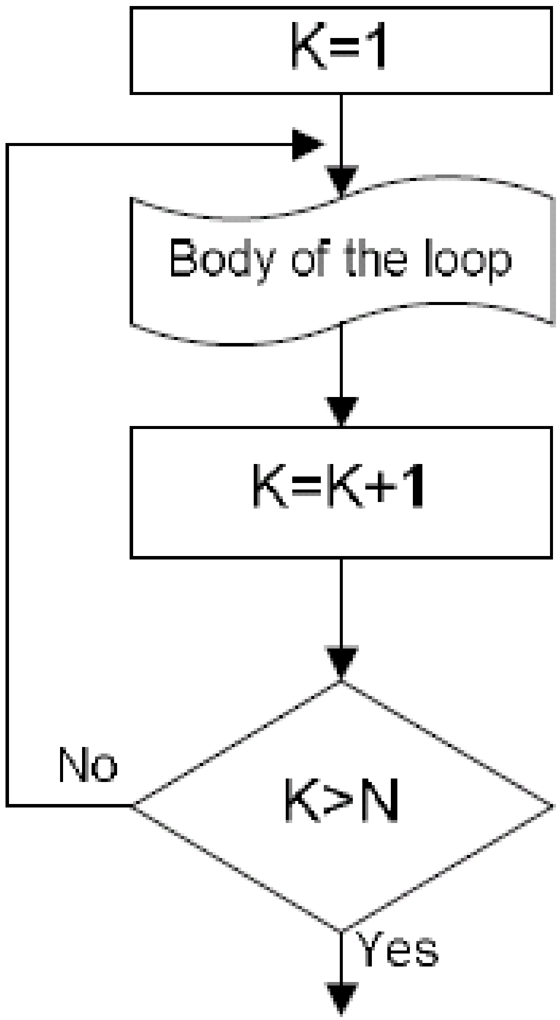
END;

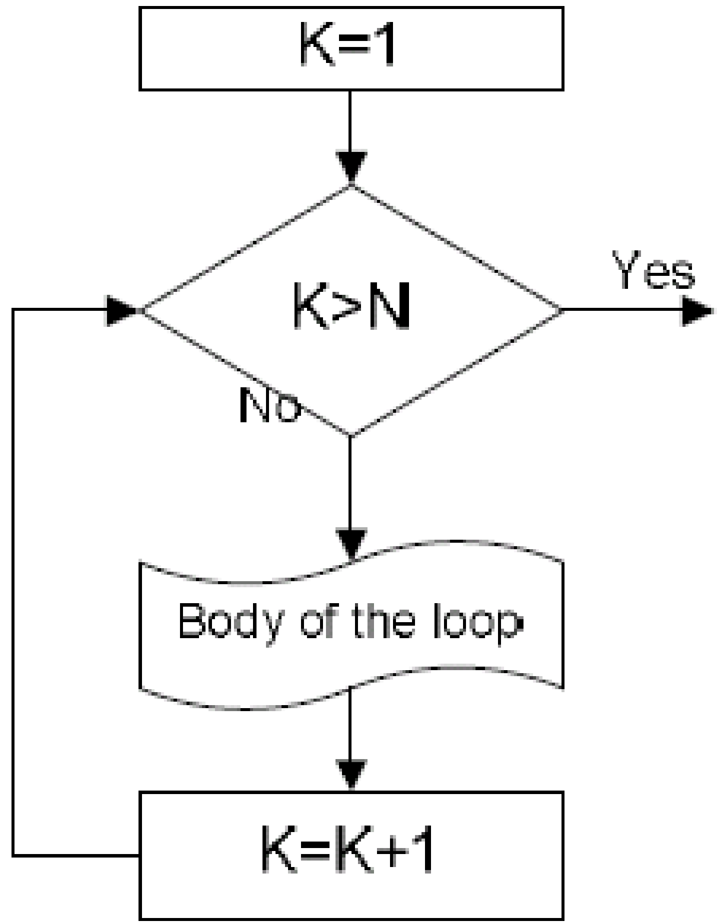
- **FORTRAN**

Do n K=1 , N

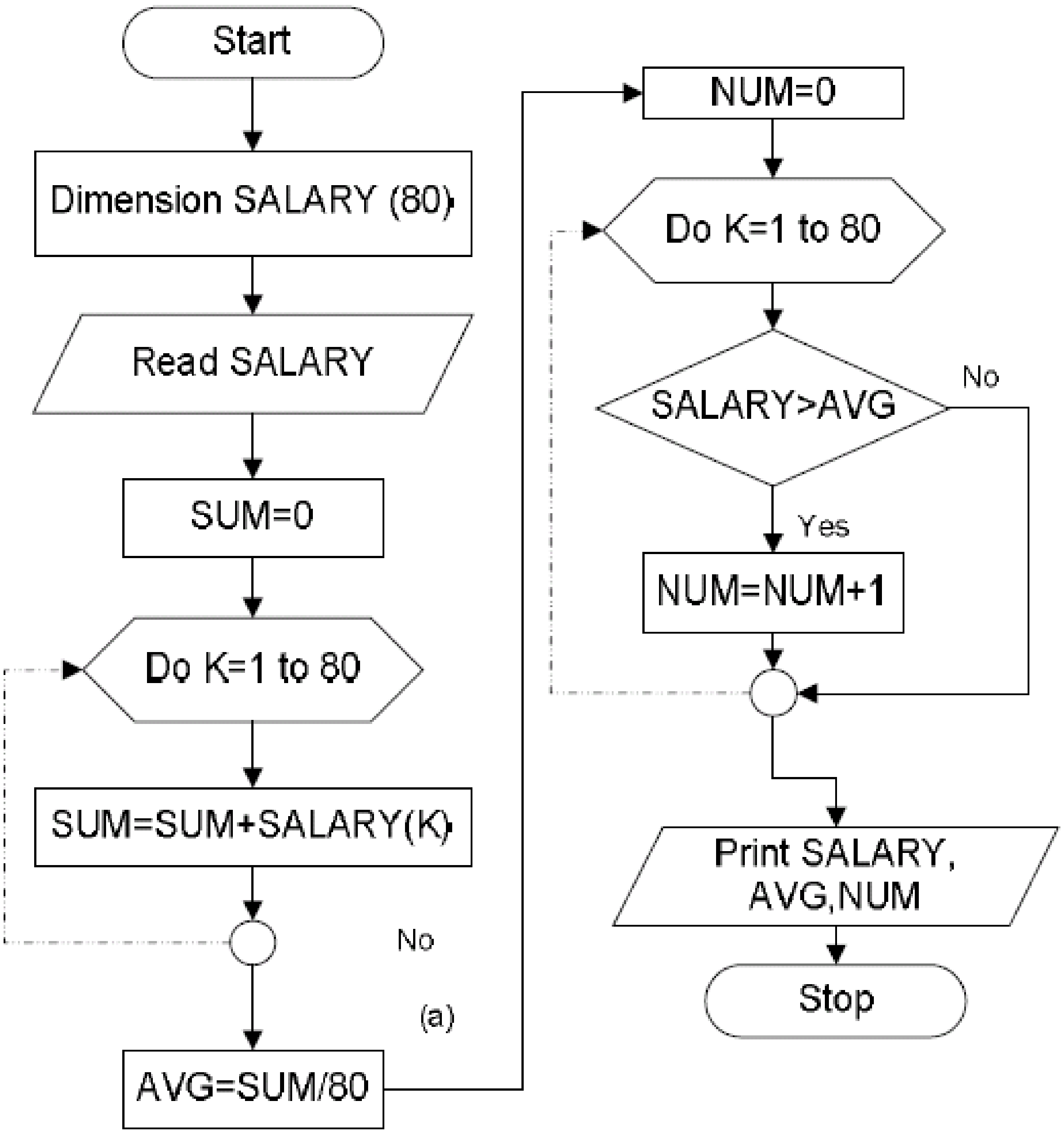
{body of loop}

n CONTIOUE


K=1
Body of the loop
K=K+1
K>N
No
Yes
(a)


K=1
K>N
Yes
No
Body of the loop
K=K+1
(b)

- **Example : A company has 80 employees give a flowchart that**
- finds the average salary and the number of employees earning above the average salary. Observe that the salaries are read into an array, SALARY. Next, the average salary, AVG, is calculated.
- Then each salary , SALARY(K), is compared with AVG to obtain the number NUM of salaries grater than AVG.


Start
Dimension SALARY (80)
Read SALARY
SUM=0
Do K=1 to 80
SUM=SUM+SALARY(K)
No
(a)
AVG=SUM/80
NUM=0
Do K=1 to 80
SALARY>AVG
No
Yes
NUM=NUM+1
Print SALARY,
AVG,NUM
Stop

PROBLEMS

- **Prob. 1.** Write an algorithm and draw a flowchart to print the square of all numbers from 1 to10.
- **Prob. 2.** Write an algorithm and draw a flowchart to print the SUM of numbers from LOW to HIGH. Test with LOW=3 and HIGH=9.
- **Prob. 3.** Write an algorithm and draw a flowchart to print all numbers between LOW and HIGH that are divisible by NUMBER.
- **Prob. 4.** Draw a flowchart for a program that reads 10 numbers from the user and prints out their sum, and their product.

- **Prob. 5.** Write an algorithm and draw a flowchart to count and print all numbers from LOW to HIGH by steps of STEP. Test with LOW=0 and HIGH=100 and STEP=5.

- **Prob. 6.** Write an algorithm and draw a flowchart to print the multiplication table for 6's. i.e.

  ---- $1 \times 6 = 6$

  ---- $2 \times 6 = 12$

  …

  ---- $12 \times 6 = 72$

- **Prob. 7.** Write an algorithm and draw a flowchart that will find and print the product of 3 numbers.

- **Prob. 8.** Write an algorithm and draw a flowchart that will find and print
- The factorial of NUMBER is FACTORIAL.
- Test the flowchart for NUMBER=5.