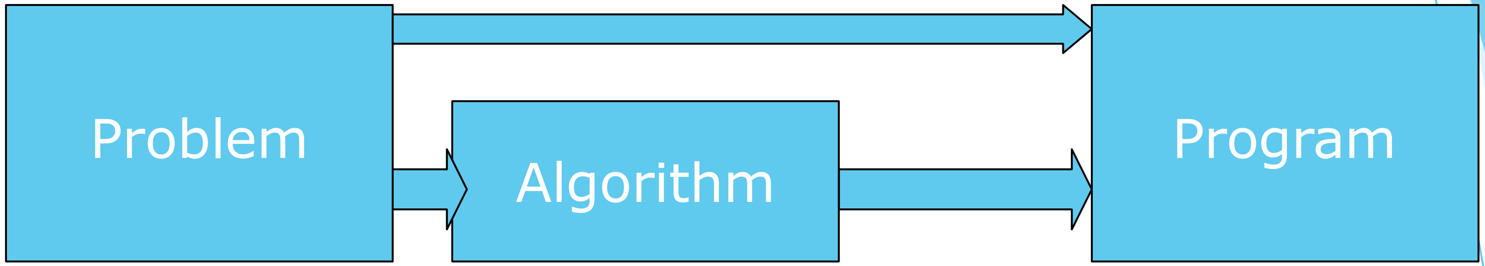
ALGORITHMS AND FLOWCHARTS

Program design

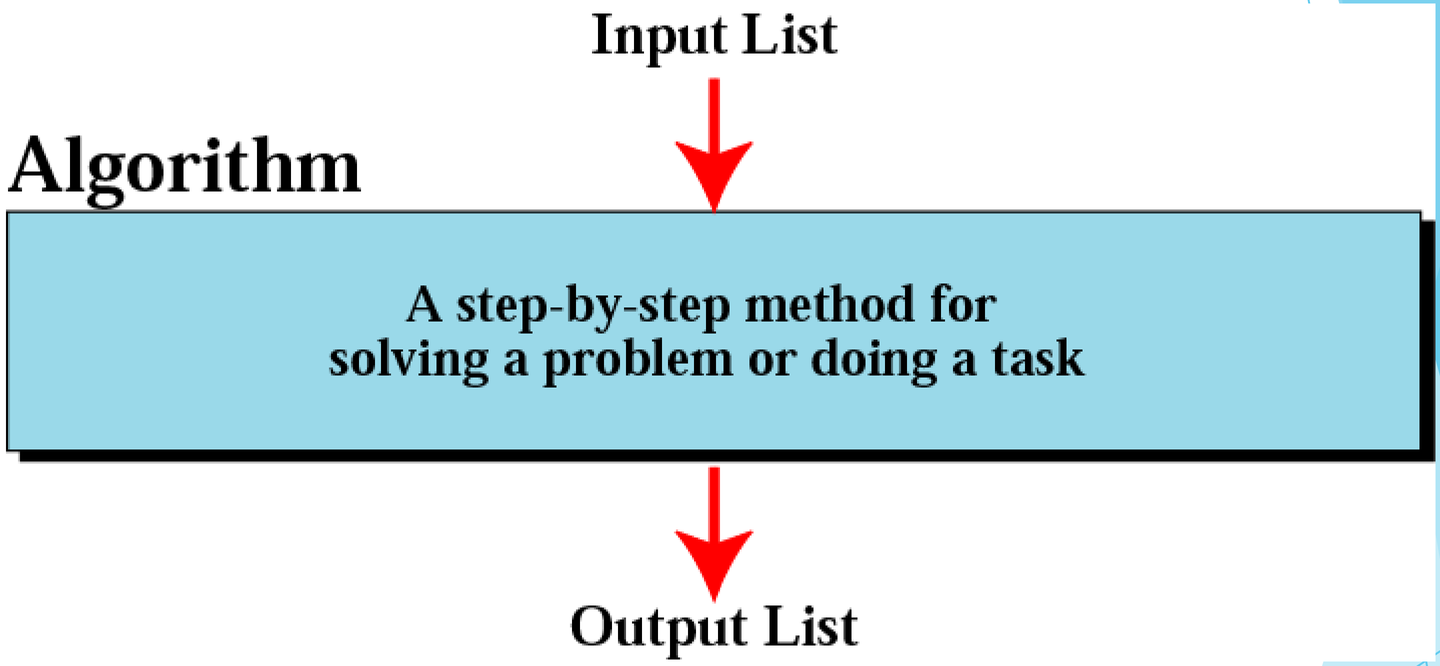


- **Program Design Process** has 2 phases:
  - **Problem Solving** Phase
    - Creates an algorithm that solves the problem
  - **Implementation** (Coding) Phase
    - Translates the algorithm into a programming language

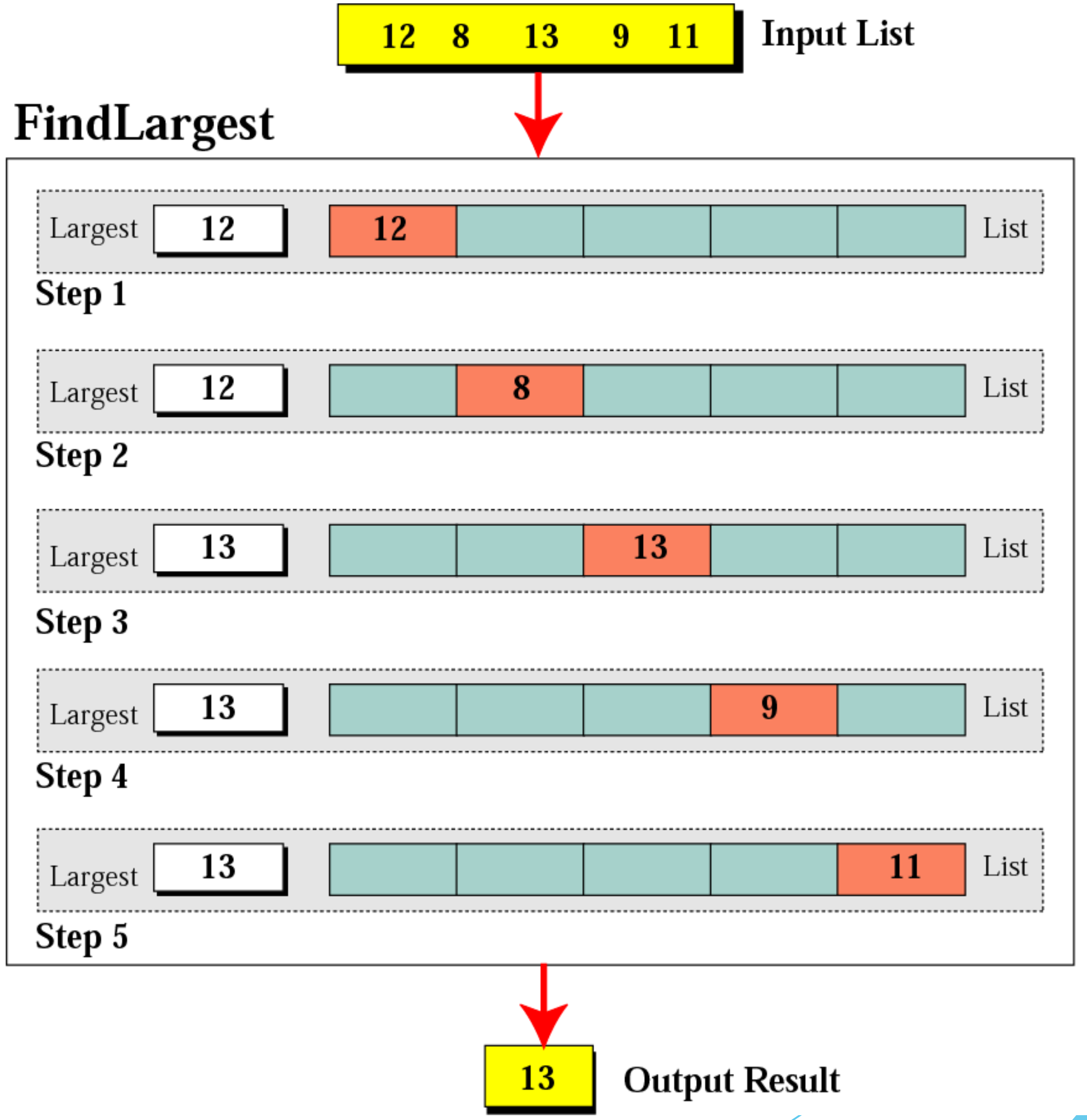
Algorithms

- An algorithm is a finite set of steps defining the solution of a particular problem.
- Need not to belong one particular language
- Sequence of English statements can also be algorithm
- It is not a computer program
- An algorithm can be expressed in English like language called pseudo code, in a programming language or in the form of flowchart.

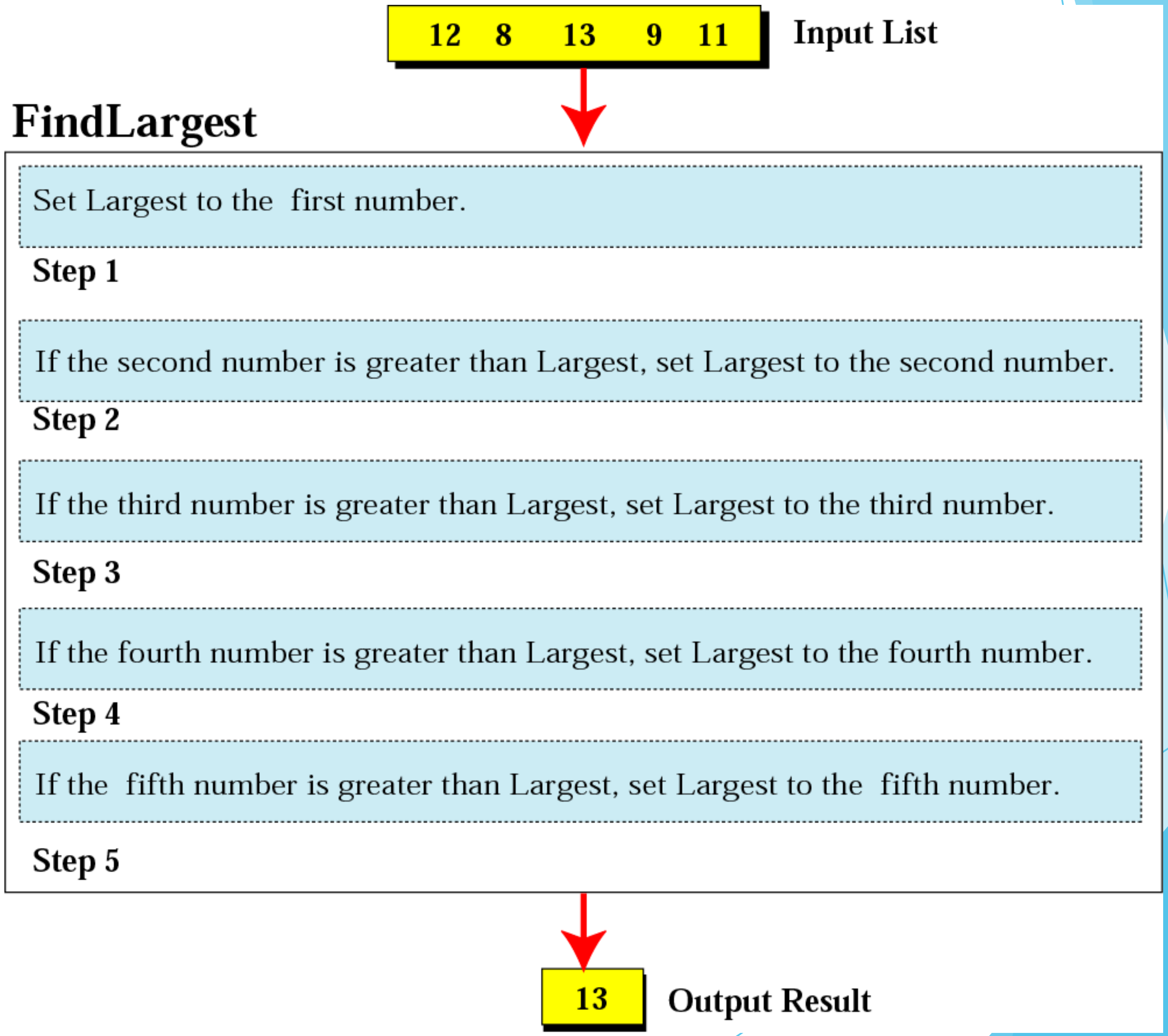
Informal definition of an algorithm



Finding the largest integer among five integers



Defining actions in Find Largest algorithm


12 8 13 9 11
Input List
FindLargest
Set Largest to the first number.
Step 1
If the second number is greater than Largest, set Largest to the second number.
Step 2
If the third number is greater than Largest, set Largest to the third number.
Step 3
If the fourth number is greater than Largest, set Largest to the fourth number.
Step 4
If the fifth number is greater than Largest, set Largest to the fifth number.
Step 5
13
Output Result

Algorithm Vs Program

- What is the difference between an algorithm and a program?

  → a program is an *implementation* of an algorithm to be run on a specific computer and operating system.

  → an algorithm is more abstract – it does not deal with machine specific details – think of it as a *method* to solve a problem.

- What is good algorithm?

Efficient algorithms are good, we generally measure efficiency of an algorithm on the basis of:

1. Time: algorithm should take minimum time to execute.
2. Space: algorithm should use less memory.

Algorithm Specification

Every algorithm must satisfy the following criteria:

- **Input.** Zero or more quantities are externally supplied.
- **Output.** At least one quantity is produced.
- **Definiteness.** Each instruction must be clear and unambiguous(Unique meaning).
- **Finiteness.** An algorithm terminates in a finite number of steps.
- **Effectiveness.** Every instruction must be basic enough to be carried out than, means not so complex.

ALGORITHM REPRESENTATION

Example 1

Write an algorithm that finds the average of two numbers

Solution:

```
Step 1: Start
Step 2: Declare variables num1, num2
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2
Step 5: Divide result by value 2
Step 6: Display result
Step 7: Stop
```

Example 2

Write an algorithm to change a numeric grade to a pass/fail grade.

Solution:

```
Pass/Fail Grade

Step 1: Start
Step 2: Define one number
Step 3: if (the number is greater than or equal to 40)
        then
                Set the grade to “pass”
        else
                Set the grade to “fail”  End if
Step 4: Display the grade
Step 5: Stop
```

Example 3

Write an algorithm for grading System by following method.

Marks range	Grade
>=80	A
>=70 & <80	B
>=60 & <70	C
>=50 & <60	D
<50	F

Algorithm for Grading

Solution

```
    Algorithm For Grade
    Input: One number
1.  if (the number is between 80 and 100, inclusive)
    then
    1. Set the grade to “A”
    End if
2.  if (the number is between 70 and 79, inclusive)
    then
    1. Set the grade to “B”
    End if
```

Continues on the next slide

```
3. if (the number is between 60  and 69, inclusive)
   then
   1. Set the grade to “C”
   End if
4. if (the number is between 50  and 59, inclusive)
   then
   1. Set the grade to “D”
   End if
5. If (the number is less than 50)
   then
   1. Set the grade to “F”
   End if
6. Return the grade
   End
```

Advantages Of Algorithm

- It provides the core solution to a given problem. This solution can be implemented on a computer system using any programming language of user’s choice.
- It facilitates program development by acting as a design document or a blueprint of a given problem solution.
- It ensures easy comprehension of a problem solution as compared to an equivalent computer program.
- It eases identification and removal of logical errors in a program.
- It facilitates algorithm analysis to find out the most efficient solution to a given problem.

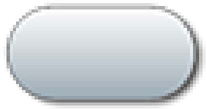
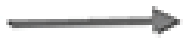
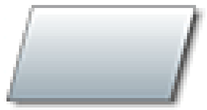
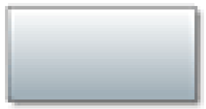
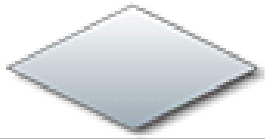
Disadvantages Of Algorithm

- In large algorithms, the flow of program control becomes difficult to track.
- Algorithms lack visual representation of programming constructs like flowcharts; thus, understanding the logic becomes relatively difficult.

Flowchart

A graphical representation of an algorithm, often used in the design phase of programming to work out the logical flow of a program.

- Visual way to represent the information flow
- Make our logic more clear
- Help during writing of program
- Make testing and debugging easy

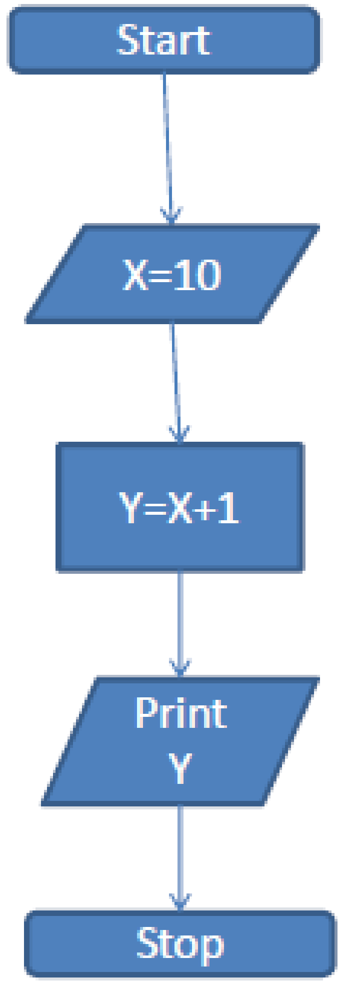
Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

Flowchart or program constructs

- **Sequence**: The order of execution, this typically refers to the order in which the code will execute. Normally code executes line by line, so line 1 then 2 then 3 and so on.
- **Selection**: Selection, like branching, is a method of controlling the execution sequence, you can create large control blocks, using **if statements** testing a condition, or **switch statements** evaluating a variable etc to control and change the execution of the program depending on this environment and changing variables.
- **Iteration** (Repetition): Iteration is typically used to refer to collections and arrays of variables and data. Repeating set of instruction. Counting from 1 to 10, you are iterating over the first 10 numbers. for, while, do-while loops will be implemented for iteration.

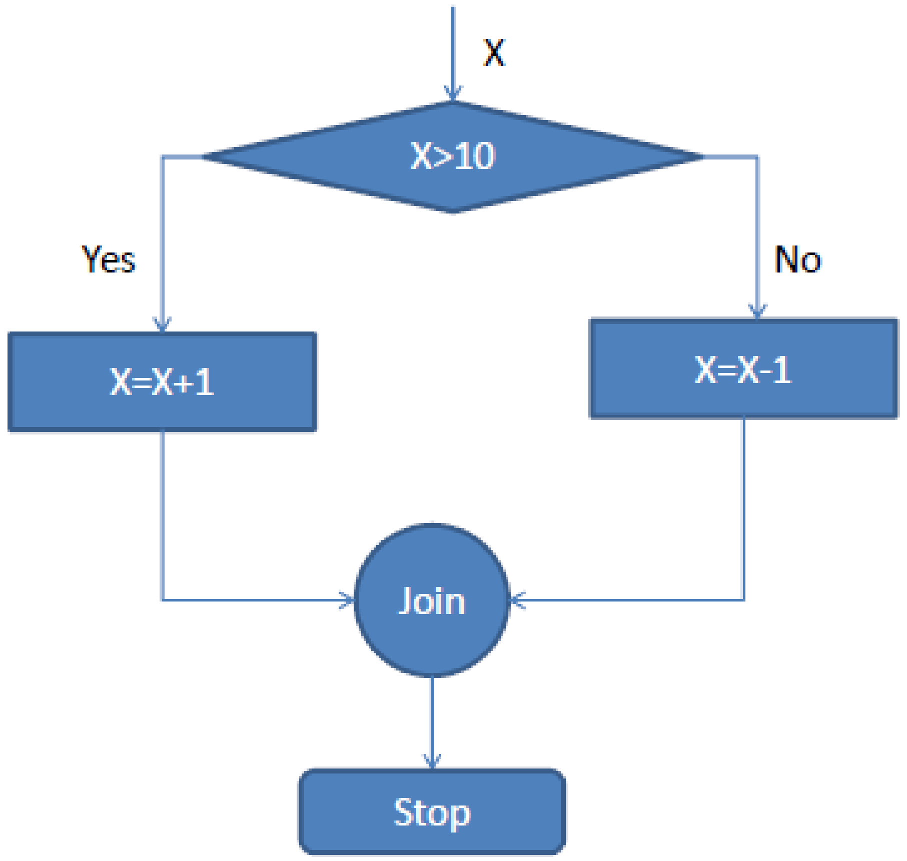
Flowchart Constructs

Sequence



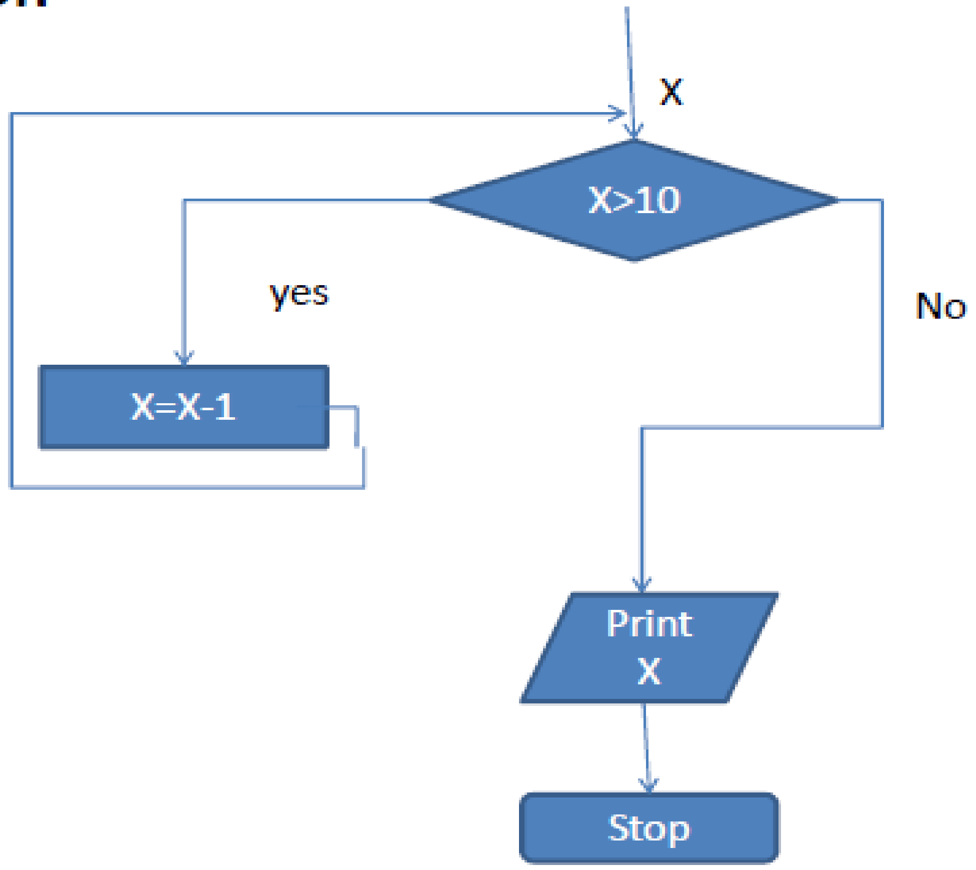
Flowchart Constructs (cont..)

Selection



Flowchart Constructs (cont..)

Repetition



Example-1

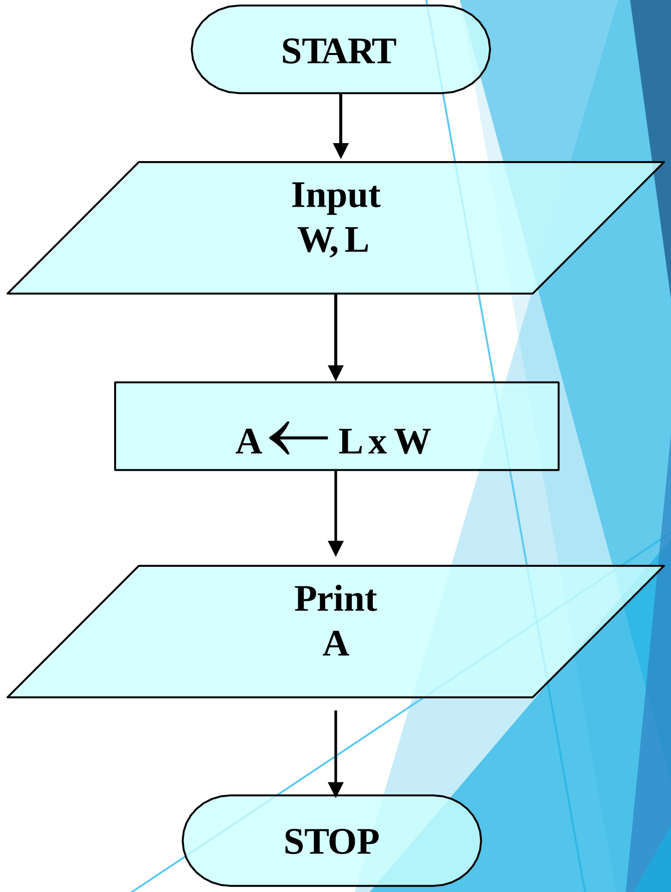
Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Algorithm

- Step 1: Start
- Step 2: Take Width and Length as input
- Step 3: Calculate Area by Width* Length
- Step 4: Print Area.
- Step 5: End

Example-1

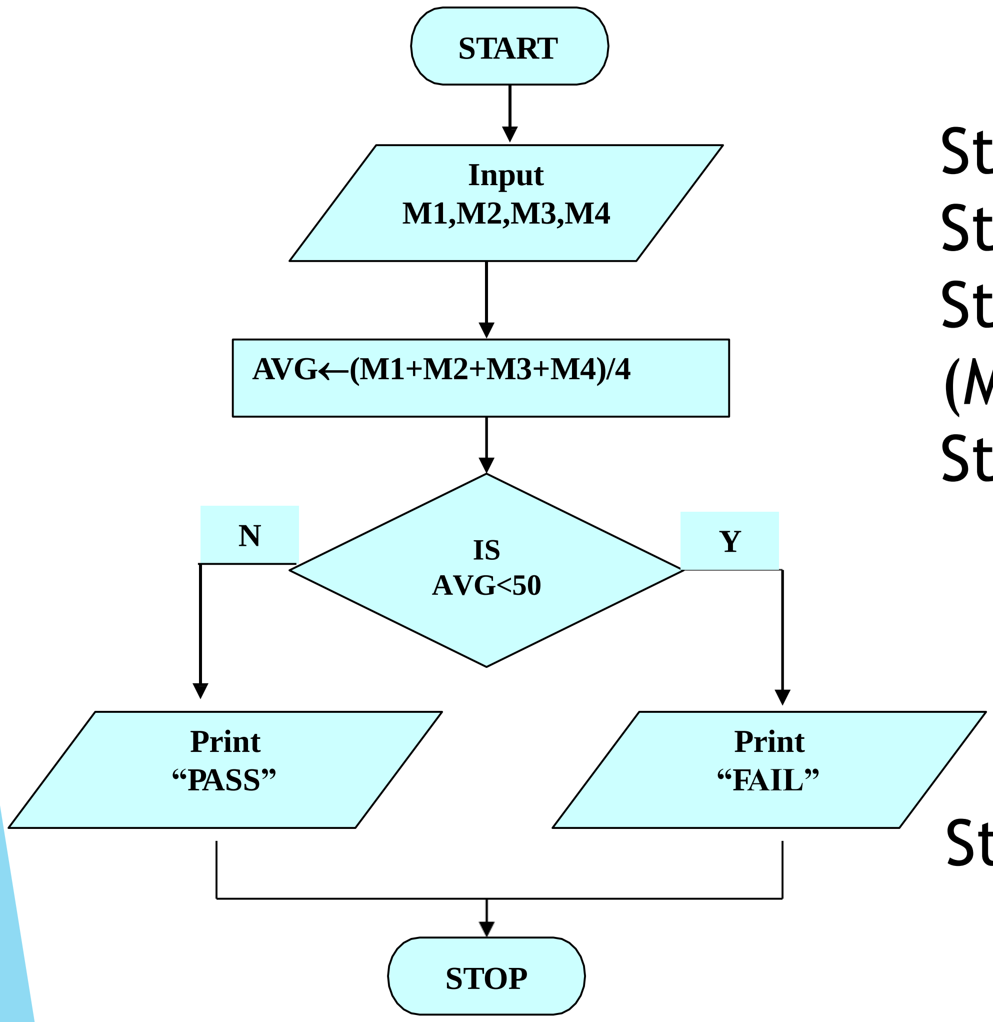
- Step 1: Start
- Step 2: Input W,L
- Step 3: A ← L x W
- Step 4: Print A
- Step 5: End



Example-2

- Write an Pseudo code and draw a flowchart that will take marks of four subjects and calculate the average. Then if average marks are greater than 50 then print PASS otherwise print FAIL.

Example-2



```
Step 1:  Start
Step 2:  Input M1,M2,M3,M4
Step 3: AVG ← (M1+M2+M3+M4)/4
Step 4:         if (AVG <50) then
                Print "FAIL"
        else
                Print "PASS"
        endif
Step 5: End
```

Example-3

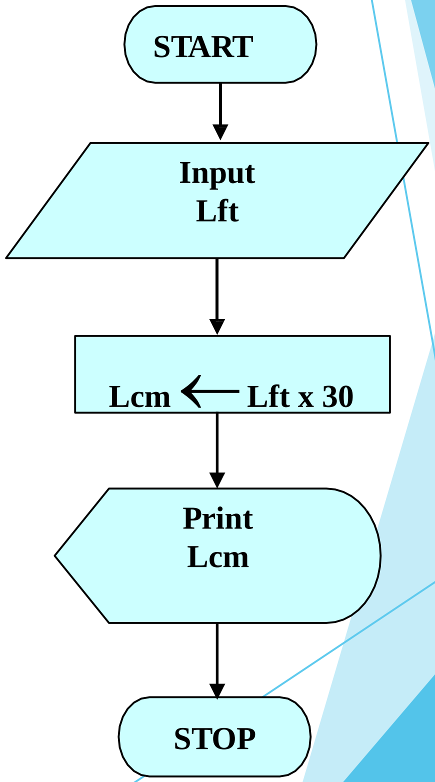
- **Write an algorithm and draw a flowchart to convert the length in feet to centimeter.**

Example-3

Algorithm

- Step 1: Start
- Step 2: Input Lft
- Step 3: Lcm $\leftarrow$ Lft x 30
- Step 4: Print Lcm
- Step 5: End

Flowchart



Example-4

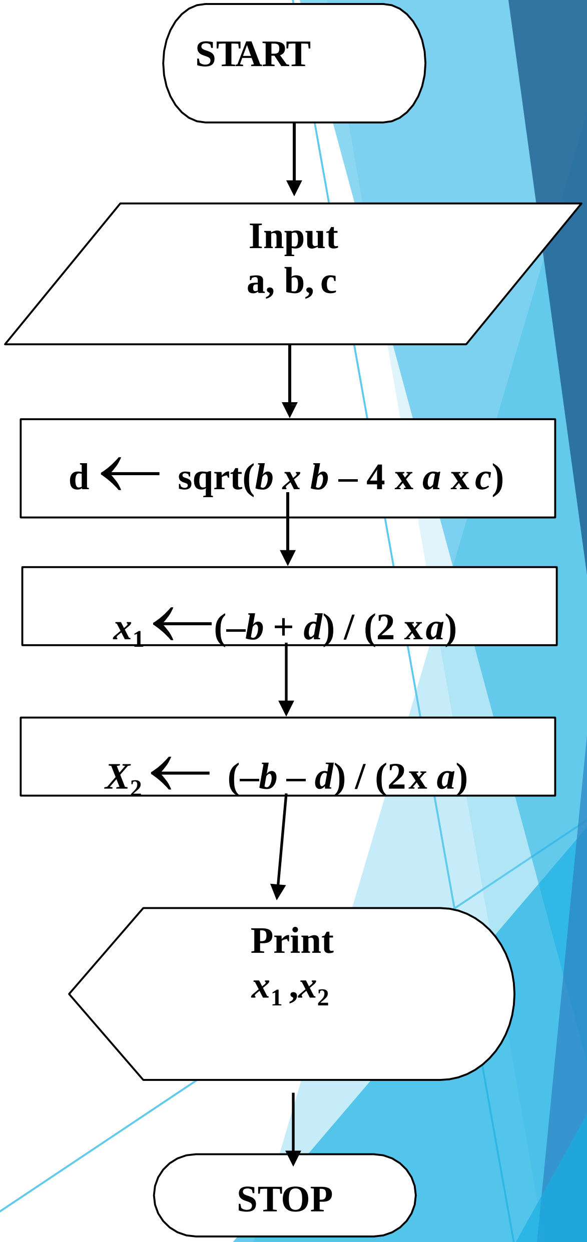
- **Write an algorithm and draw a flowchart that will calculate the roots of a quadratic equation** $ax^2 + bx + c = 0$

- **Hint: d = sqrt (** $b^2 - 4ac$ **), and the roots are: *x*1 = (-*b* + *d*)/2*a* and *x*2 = (-*b* - *d*)/2*a***

Example-4

Algorithm:

- Step 1: Start
- Step 2: Input a, b, c
- Step 3: $d \leftarrow$ sqrt ($b \times b - 4 \times a \times c$)
- Step 4: $x1 \leftarrow (-b + d) / (2 \times a)$
- Step 5: $x2 \leftarrow (-b - d) / (2 \times a)$
- Step 6: Print $x1$, $x2$
- Step 7: End


START
Input
a, b, c
$d \leftarrow \text{sqrt}(b \times b - 4 \times a \times c)$
$x_1 \leftarrow (-b + d) / (2 \times a)$
$X_2 \leftarrow (-b - d) / (2 \times a)$
Print
x_1, x_2
STOP

Flow Chart`s Limitation

- For very large program, flow chart goes for many pages
- Costly to draw flow charts for large program
- Difficult to modify

Thank you!