SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMMING FOR PROBLEM SOLVING

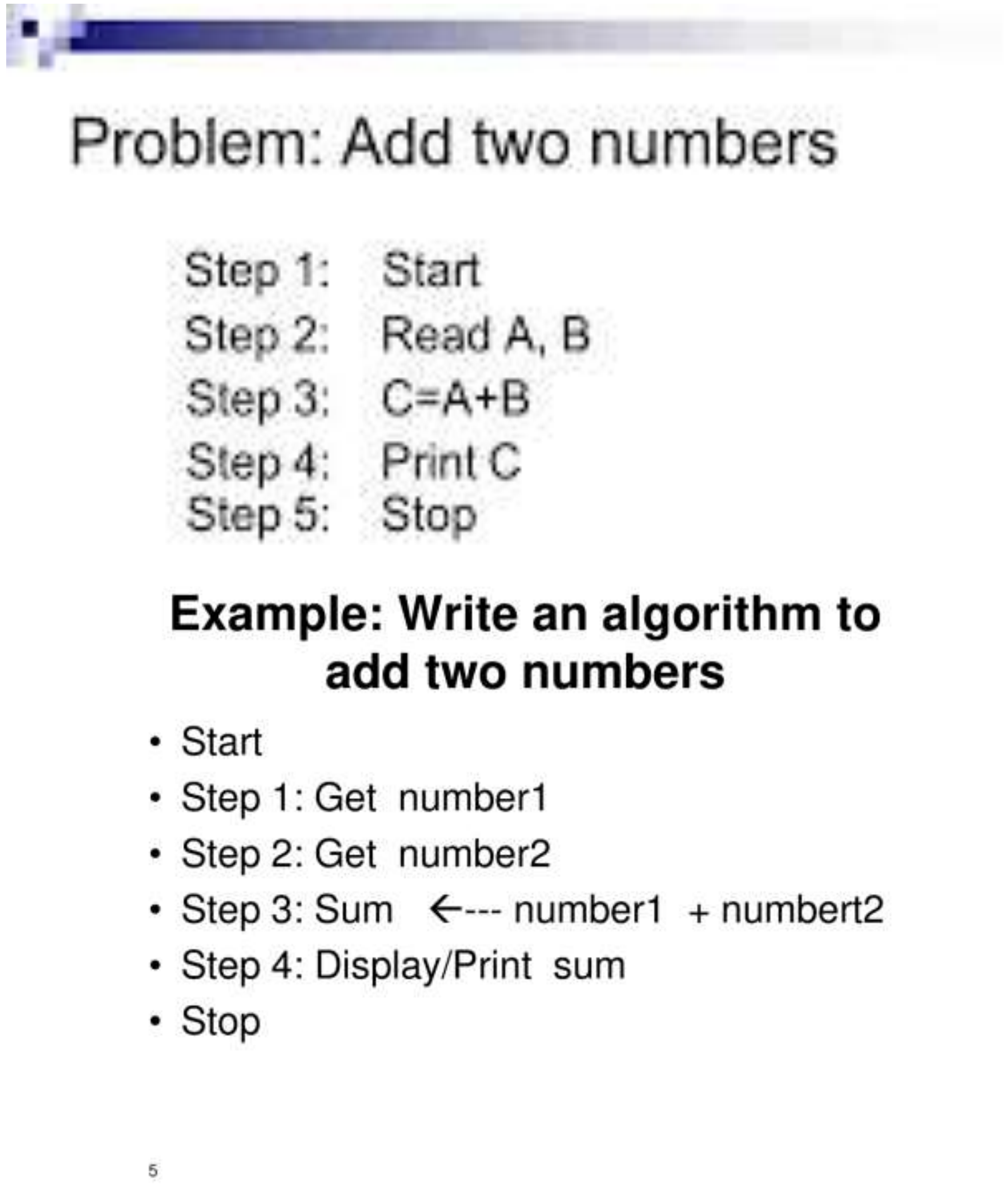
I YEAR - I SEM

UNIT 1 – Introduction to Problem Solving Techniques

TOPIC 4 – Building Blocks of Algorithm

ALGORITHM

- It is defined as a sequence of instructions that describe a method for solving a problem.
- In other words it is a step by step procedure for solving a problem.
  - Should be written in simple English
  - Each and every instruction should be precise and unambiguous.
  - Instructions in an algorithm should not be repeated infinitely.
  - Algorithm should conclude after a finite number of steps.
  - Should have an end point
  - Derived results should be obtained only after the algorithm terminates.



Problem: Add two numbers

Step 1: Start
Step 2: Read A, B
Step 3: C=A+B
Step 4: Print C
Step 5: Stop

Example: Write an algorithm to add two numbers

- Start
- Step 1: Get number1
- Step 2: Get number2
- Step 3: Sum ←--- number1 + numbert2
- Step 4: Display/Print sum
- Stop

QUALITIES OF A GOOD ALGORITHM

- The following are the primary factors that are often used to judge the quality of the algorithms.
- Time – To execute a program, the computer system takes some amount of time. The **lesser** is the time required, the better is the algorithm.
- Memory – To execute a program, computer system takes some amount of memory space. **The lesser** is the memory required, the better is the algorithm.
- Accuracy – Multiple algorithms may provide suitable or correct solutions to a given problem, some of these may **provide more** accurate results than others, and such algorithms may be suitable

Example

Write an algorithm to print „Good Morning"

Step 1: Start

Step 2: Print "Good Morning"

Step 3: Stop

BUILDING BLOCKS OF ALGORITHM

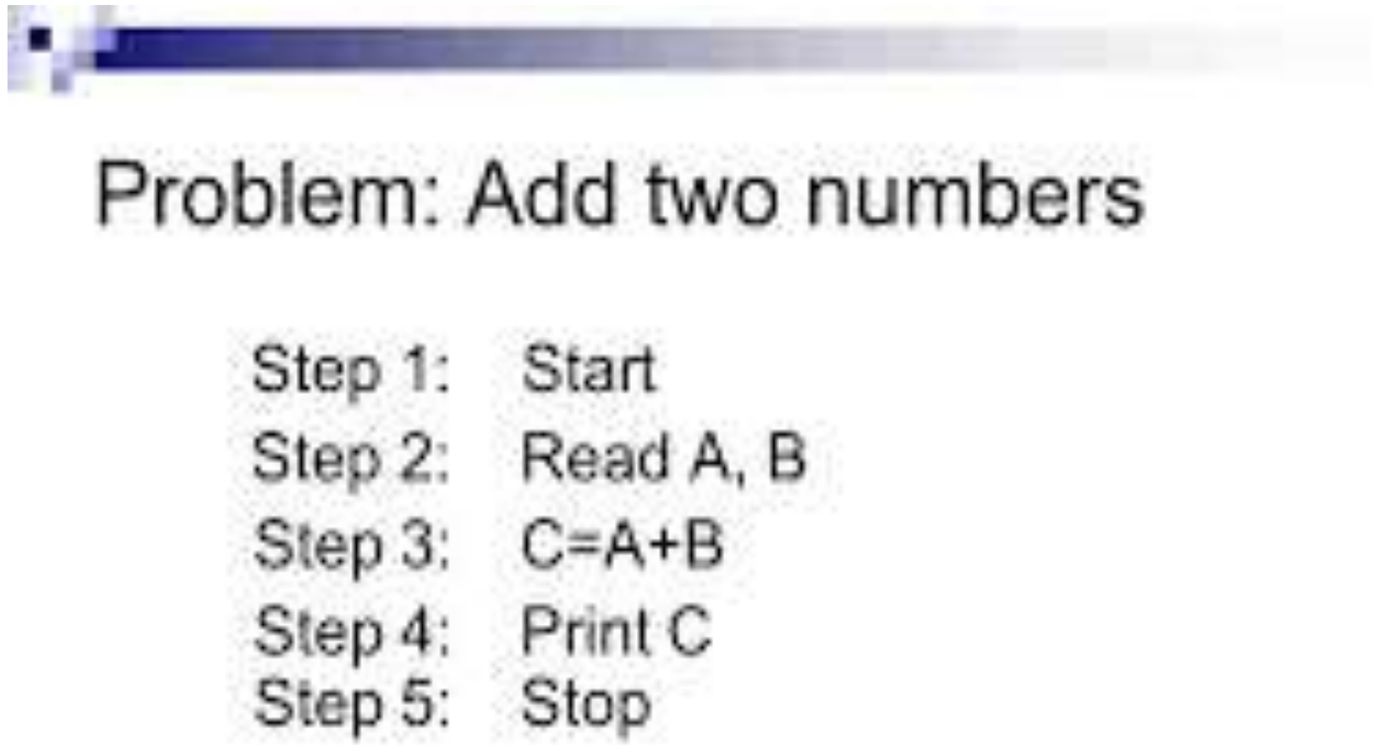
- As algorithm is a part of the blue-print or plan for the computer program.
- An algorithm is constructed using following blocks.
  - Statements
  - States
  - Control flow
  - Function

STATEMENTS

- Statements are **simple sentences** written in algorithm for specific purpose.
- Statements may consists of assignment statements, **input/output statements**, comment statements

- Statements might include some of the following actions
  - **input** data-information given to the program
  - **process** data-perform operation on a given input
  - **output dat**a - processed result

- Example:
  - · Read the value of ‘a’ //This is input statement
  - · Calculate c=a+b //This is assignment statement
  - · Print the value of c // This is output statement
  - . Comment statements are given after // symbol, which is used to tell the purpose of the line.

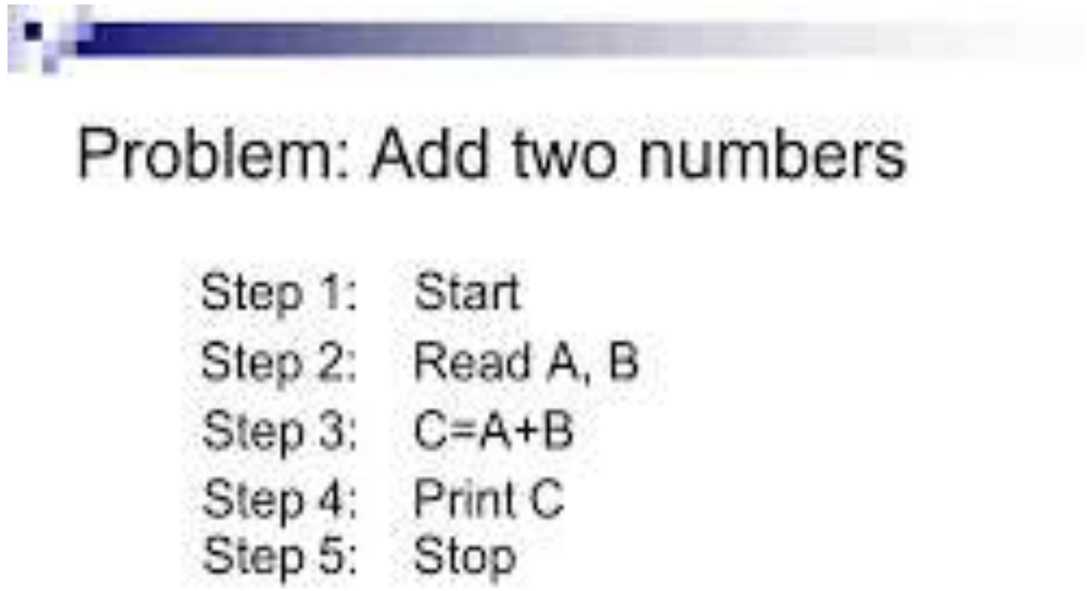


Problem: Add two numbers

```
Step 1:   Start
Step 2:   Read A, B
Step 3:   C=A+B
Step 4:   Print C
Step 5:   Stop
```

STATES

- An algorithm is deterministic **automation** for accomplishing a goal which, given an initial state, will terminate in a defined end-state.
- In other words, **Transition from one process to another process** under specified condition with in a time is called state.
- An algorithm will definitely have **start state and end state**



Problem: Add two numbers

```
Step 1:   Start
Step 2:   Read A, B
Step 3:   C=A+B
Step 4:   Print C
Step 5:   Stop
```

CONTROL FLOW

- Control flow which is also stated as flow of control, determines what section of code is to run in program at a given time.

- There are three types of flows, they are
  - 1. Sequential control flow
  - 2. Selection or Conditional control flow
  - 3. Looping, iteration or repetition control flow

SEQUENTIAL CONTROL FLOW

- Sequential control structure is used to perform the **action one after another**.
- **Only one step** is executed once.
- The logic is **top to bottom** approach.

- Example

Description: To find the sum of two numbers.

```
STEP 1. Start
STEP 2. Read the value of ‘a’
STEP 3. Read the value of ‘b’
STEP 4. Calculate sum=a+b
STEP 5. Print the sum of two number
STEP 6. Stop
```

SELECTION OR CONDITIONAL CONTROL FLOW

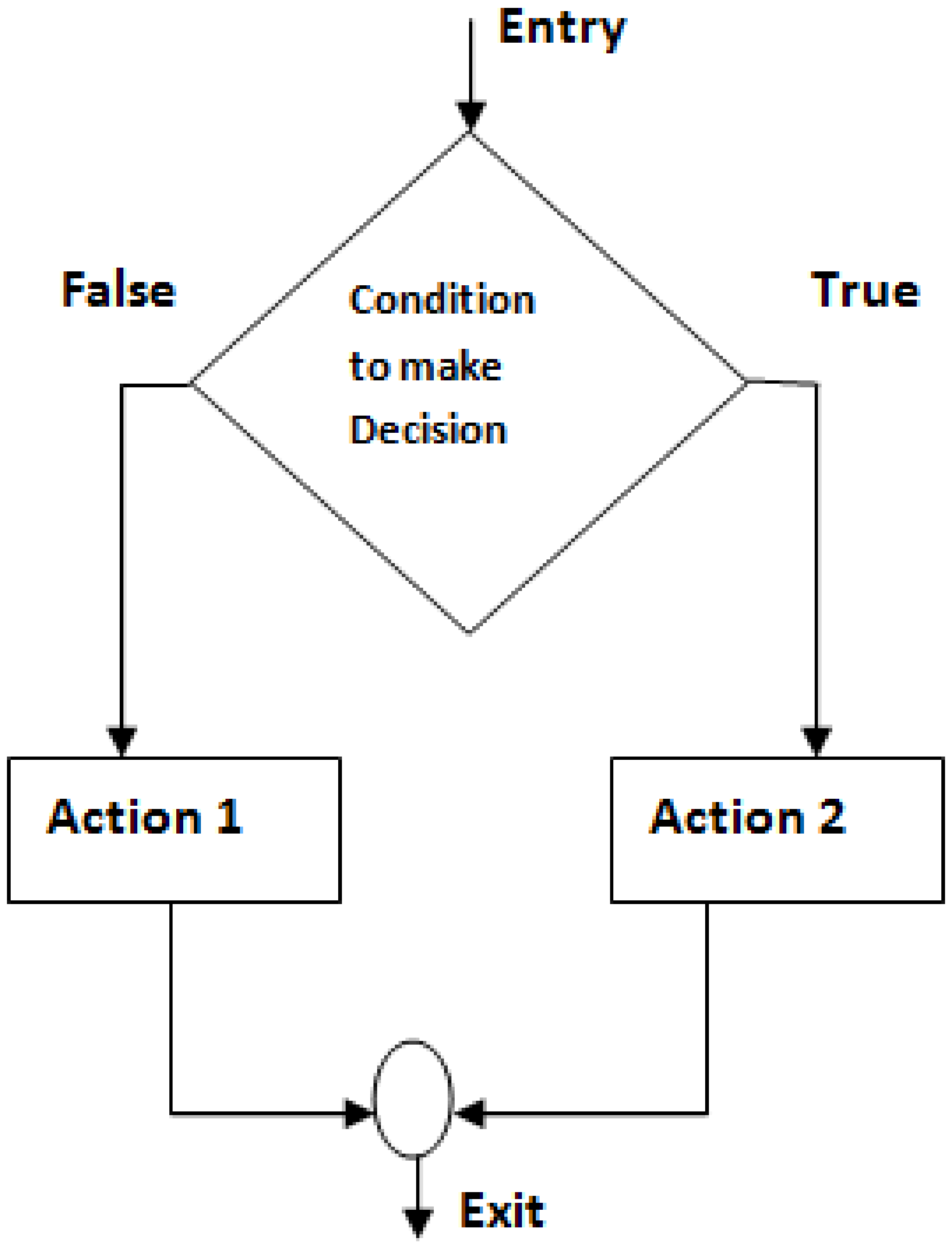
- Selection flow allows the program to make **“choice” between two alternate paths** based on condition.
- It is also called as **decision structure.**

Basic structure:

```
IFCONDITION is TRUE then
          perform some action
ELSE IF CONDITION is FALSE then
          perform some action
```

Example

```
//Description: finding the greater number
STEP 1. Start
STEP 2. Read a
STEP 3. Read b
STEP 4. If a>b then
          STEP 4.1. Print a is greater
                    else
          STEP 4.2. Print b is greater
STEP 5. Stop
```



REPETITION CONTROL FLOW

- Repetition control flow means that **one or more steps are performed repeatedly** until some **condition** is reached.
- This logic is used for producing **“loops”** in program logic when one or more instructions may need to be executed several times depending on condition.

Basic Structure:

```
Repeat untilCONDITIONis true
        Statements
```

Example

```
//Description: to print the values from 1 to n
STEP 1. Start
STEP 2. Read the value of ‘n’
STEP 3. Initialize i as 1
STEP 4. Repeat step 4.1 until i< n
        STEP 4.1. Print i
STEP 5. Stop
```

FUNCTION

- A function is a **block** of organized, reusable code that is used to perform a single, related action.
- Function is also named as methods, sub-routines.
- For complex problems, the problem is been divided into **smaller and simpler tasks** during algorithm design

Benefits of Using Functions

- Reduction in line of code
- Code reuse
- Better readability
- Information hiding
- Easy to debug and test
- Improved maintainability

<u>Basic Syntax</u>

```
function_name(parameters)
        function statements
end function
```

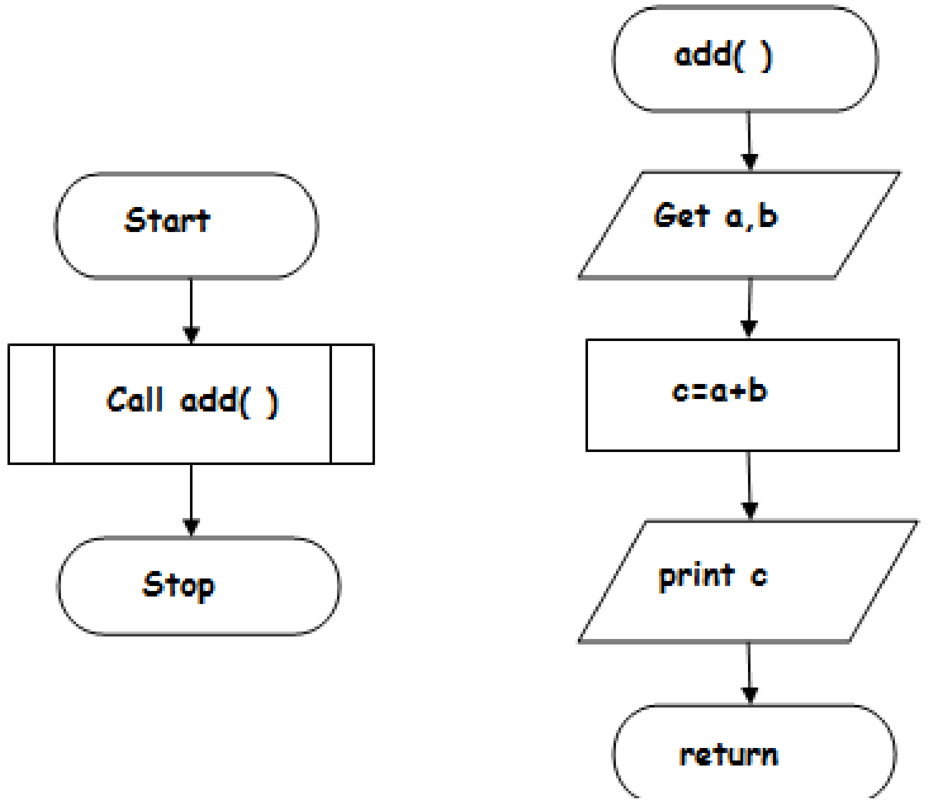
Algorithm for addition of two numbers using function

<u>Main function()</u>

```
Step 1: Start
Step 2: Call the function add()
Step 3: Stop
```

<u>sub function add()</u>

```
Step 1: Function start
Step 2: Get a,bValues
Step 3: add c=a+b
Step 4: Print c
Step 5: Stop
```



EXAMPLES

Problem 1:

Find the area of a Circle of radius r.

Inputs to the algorithm:

Radius r of the Circle.

Expected output:

Area of the Circle

Algorithm:

```
Step 1: Start
Step2: Read input the Radius r of the Circle
Step3: Area = PI*r*r // calculation of area
Step4: Print Area
Step 5: Stop
```

Problem2:

Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:

First num1.

Second num2.

Expected output:

Sum of the two numbers.

Algorithm:

```
Step 1: Start
Step 2: Read\input the first num1.
Step 3: Read\input the second num2.
Step 4: Sum =  num1+num2 // calculation of sum
Step 5: Print Sum
Step 6: Stop
```

EXAMPLES

Problem 3:

Convert temperature Fahrenheit to Celsius

```
        Inputs to the algorithm:
                Temperature in Fahrenheit
        Expected output:
                Temperature in Celsius
```

Algorithm:

```
Step 1: Start
Step 2: Read Temperature in Fahrenheit F
Step 3: C = 5/9*(F-32)
Step 4: Print Temperature in Celsius: C
Step 5: End
```

Problem 4:

Find the largest number between A and B

```
        Inputs to the algorithm:
                A, B
        Expected output:
                Largest A or B
```

Algorithm:

```
Step 1: Start
Step 2:Read A, B
Step 3: If A is less than B, then
        Big=B
        Small=A
        Print A is largest
Else
        Big=A
        Small = B
Step 4: Write (Display) BIG, SMALL
Step 5: Stop
```

EXAMPLES

Problem 5:

To determine a student’s average grade and indicate whether successful or fail.

```
Step 1: Start
Step 2: Input mid-term and final
Step 3: average=(mid-term + final)/2
Step 4: if (average < 60) then
                Print “FAIL”
        else
                Print “SUCCESS”
Step 5: Stop
```

Problem 6:

A algorithm to find the largest value of any three numbers.

```
Step 1: Start
Step 2: Read/input A,B and C
Step 3: If (A>=B) and (A>=C) then Max=A
Step 4: If (B>=A) and (B>=C) then Max=B
Step 5:If (C>=A) and (C>=B) then Max=C
Step 6: Print Max
Step 7: End
```