HO CHI MINH CITY UNIVERSITY OF TRANSPORT
FACULTY OF INFORMATION TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT

CHAPTER 3
PROBLEM SOLVING WITH COMPUTER

CONTENTS

1. Introduction to Problem Solving

- A computer **cannot solve a problem on its own.**
- Provide step by step solutions of the problem to the computer.
- In order to solve a problem by the computer, one has to pass though certain stages:
  1. ***Understanding the problem***
  2. ***Analyzing the problem***
  3. ***Developing the solution***
  4. ***Coding and implementation***

Understanding the problem

- State the problem clearly and unambiguously to understand exactly:
  - What the problem is?
  - What is needed to solve it?
  - What the solution should provide
  - If there are constraints and special conditions

Analyzing the problem

- In the analysis phase, we should identity the following:
  - **Inputs**: To the problem, their form and the input media to be used.
  - **Outputs**: Expected from the problem, their form and the output media to beused.
  - Special **constraints** or (necessity) **conditions** (if any).
  - Formulas or equations to be used.
- ***Example***: Compute and display the total cost of apples given the number of kilograms (kg) of apples purchased and the cost per kg of apples.
  - *Input*: Quantity of Apples purchased (in kg) and Cost per kg of Apples.
  - *Output*: Total cost of Apples (in Rs.).
  - *Constraint*: N/A
  - *Formula*: Total cost = Cost per kg x Quantity

Developing the solution

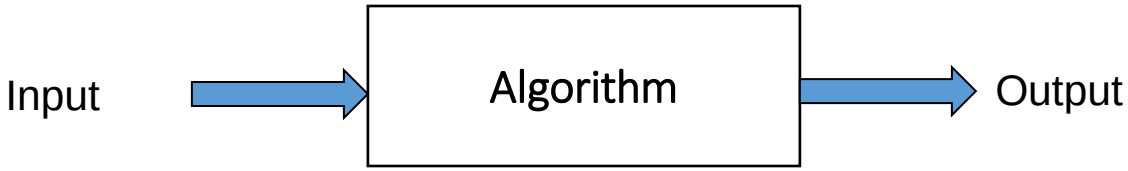
- Form a detailed step by step solution to the problem - Algorithm .
- Need to design and verify algorithm.
- Difficult problem → Use top-down design (**Divide and Conquer** approach)
  - List the major step
    - Get the data.
    - Perform the computations.
    - Display the result.
  - Perform algorithm **refinement** the step(s) may need to be broken down into a more detailed list of steps.
  - Verify that the algorithm works as intended – perform desk check

Coding and implementation

- Conversion of the detailed sequence of operations in to a language that the computer can understand.
- The process of implementing an algorithm by writing a computer program.

2. Algorithm

- **Definition**:
  - An **algorithm** is procedure consisting of a finite set of unambiguous rules (instructions) which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems for any allowable set of input quantities (if there are inputs).



  - The most important factor in the choice of algorithm is the time requirement to execute it. → Least time when executed is considered the best.

Algorithm

- **Example**:
  - Problem: Find greatest common divisor (GCD) of two integers A, B
    - Input: A, B
    - Output: GCD of A and B
  - ***Euclidean Algorithm***:
    1. Let A and B be integers with $A > B \geq 0$. If $A < B$ → swap A and B.
    2. If $B = 0$, then the GCD is A → stop.
    3. Otherwise,
       - Find remainder $R = A \bmod B$, where $0 \leq R < B$
       - Assign $A = B$, $B = R$
       - Go to step 2.

Properties of Algorithm

An algorithm must possess the following properties:

- **Finiteness**
  - An algorithm must terminate in a finite number of steps.
- **Definiteness**
  - Each step of the algorithm must be precisely and unambiguously defined.
- **Effectiveness**
  - Each step must be performed exactly in a finite amount of time.
  - Provide the correct answer to the problem.
- **Efficency**
  - Time: how fast the algorithm runs
  - Space: how much extra memory it uses
- **Generality**:
  - The algorithm can be used to solve problems of a specific type for any input data.
- **Input/output**:
  - Each algorithm must take zero, one or more quantities as input data produce one or more output values

3. Express an Algorithm

- Algorithms can be expressed in many kinds of notation, including:
  - Natural language
  - Pseudo-Code
  - Flowchart

Natural language

- ***Advantages***
  - It comes so naturally to us and can convey the steps of an algorithm to a wide audience.
  - Simple, no notation or rule.
- ***Disadvantages***
  - Tend to verbose and ambigous
  - Have no imposed structure → makes it difficult for others to follow the algorithm and feel confident in its correctness.

Pseudo-Code

- A mixture of natural language and programming language-like constructs
- There is no standard convention for writing pseudo-code; each author may have his own style.
- Pseudo-Code cannot be compiled nor executed on a computer
- ***Advantages***
  - More precise than natural language.
  - Easy to read, understand, modify
  - Converting a pseudocode to a programming language is easy
- ***Disadvantages***
  - A graphic representation of program logic is not available
  - No standard rules for using a pseudocode → communication problem occurs due to lack of standardization

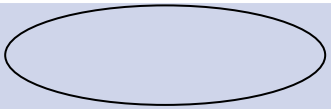
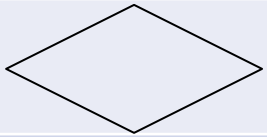
Pseudo-Code

- ***Example***: Compute the average of 10 numbers

```
1. Total=0
2. Average=0
3. For 1 to 10
4. Read number
5. Total=Total+number
6. End for
7. Average=Total/10
```

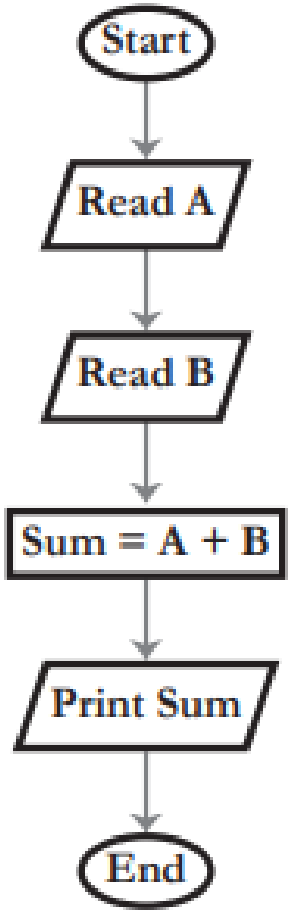
Flowchart

- The flowchart is a diagram which visually presents the flow of data through processing systems.
- Using geometrical symbols to represent the steps of the solution

Symbols	Function
	Beginning and ending points of an algorithm
	Used for any Input / Output operation
	Indicate any set of processing operation such as for storing arithmetic operations
	Indicate condition/decision
	Used to invoke a subroutine or an Interrupt program
	Shows direction of flow

Flowchart

- **Example**: find sum of two numbers
- **Advantages**:
  - Shows the logic of a problem → easier checking of an algorithm
  - Good means of communication to other users
  - Proper Debugging
- **Disadvantages**:
  - The problems are quite complicated → flowchart becomes complex and clumsy
  - If alterations are required the flowchart may require re-drawing completely

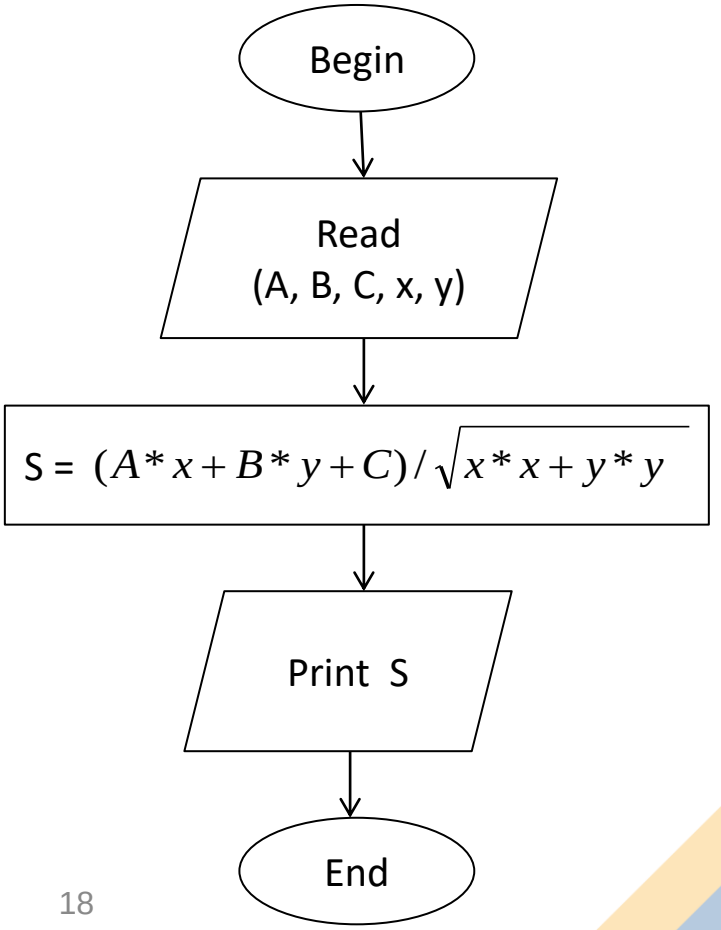


4. Basic Control Structures

- Any algorithm can be described using only three control structures:
  1. Sequence
  2. Branching (Selection)
  3. Loop (Repetition)

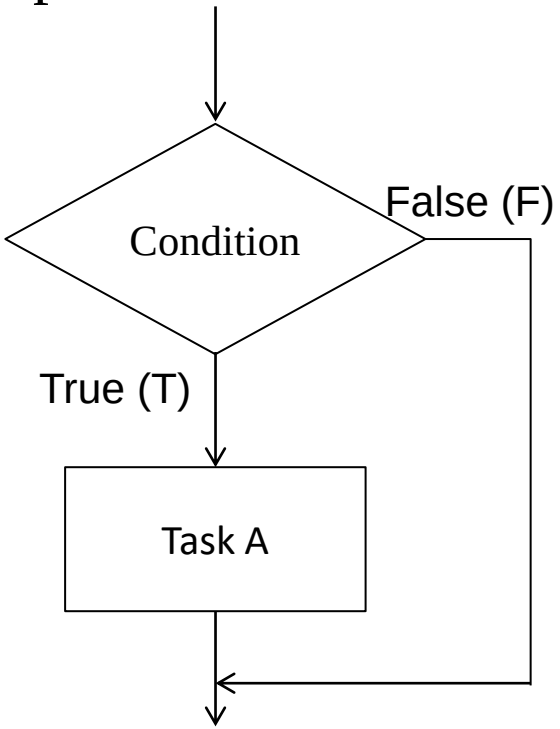
Sequence Structure

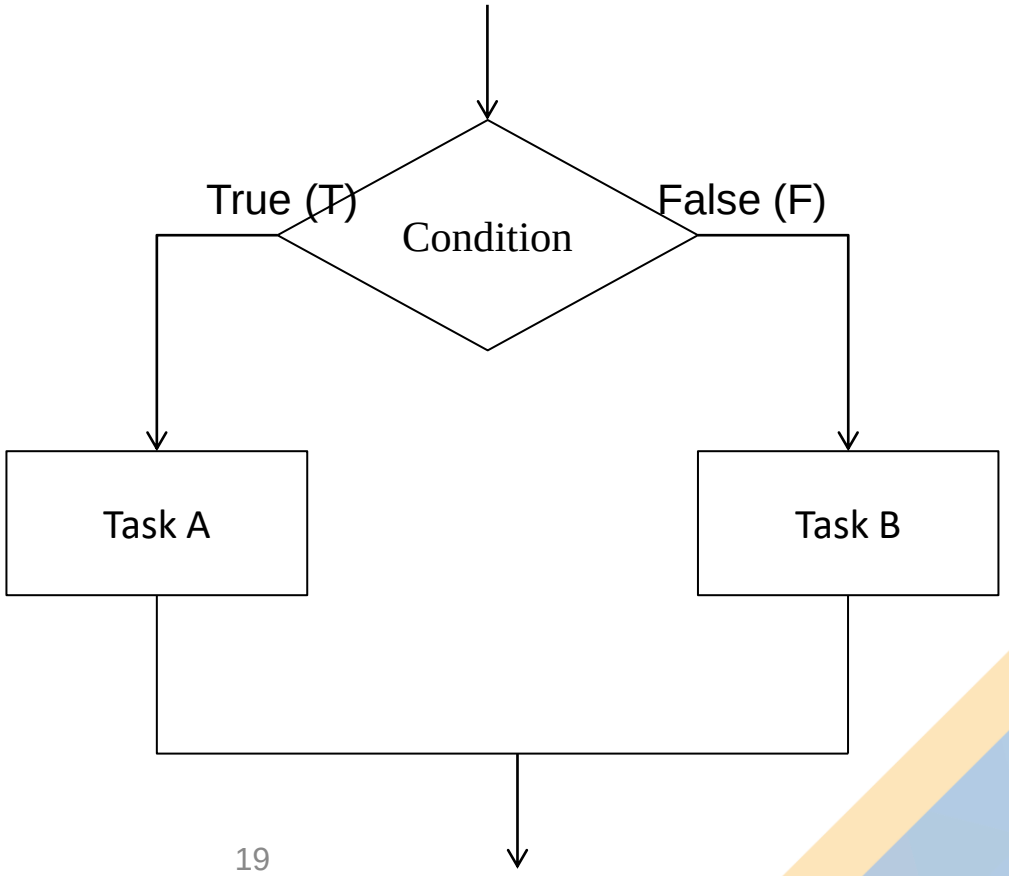
- The steps of an algorithm are carried out in a sequential manner where each step is executed exactly once
- Example:
  - Caculate S = $\frac{Ax + By + C}{\sqrt{x^2 + y^2}}$
  - Where x, y $\neq 0$



Branching Structure

- A binary decision based on some condition
- A condition is an expression that is either true (or) false
- Once the condition is evaluated, the control flows into one of two paths.

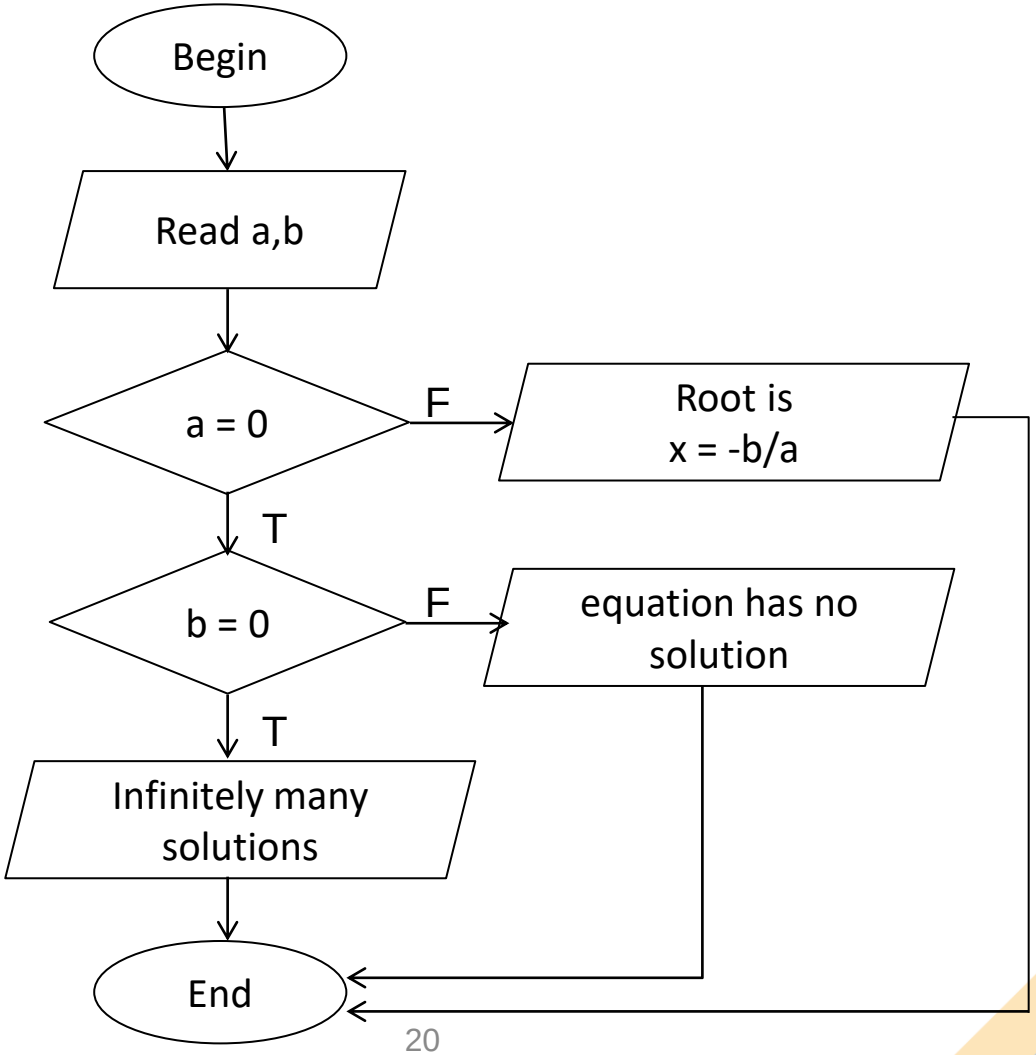




Branching Structure

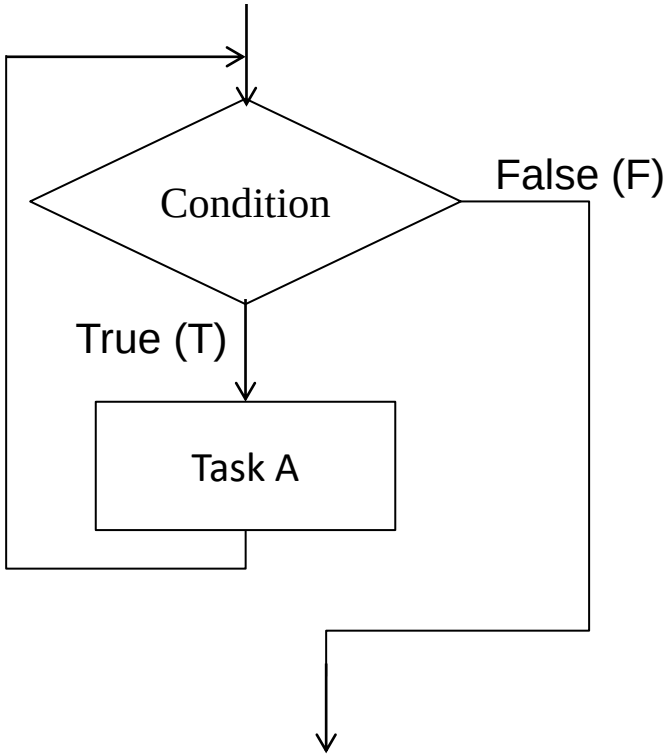
- **Example**: Draw the flowchart to find root of linear equation

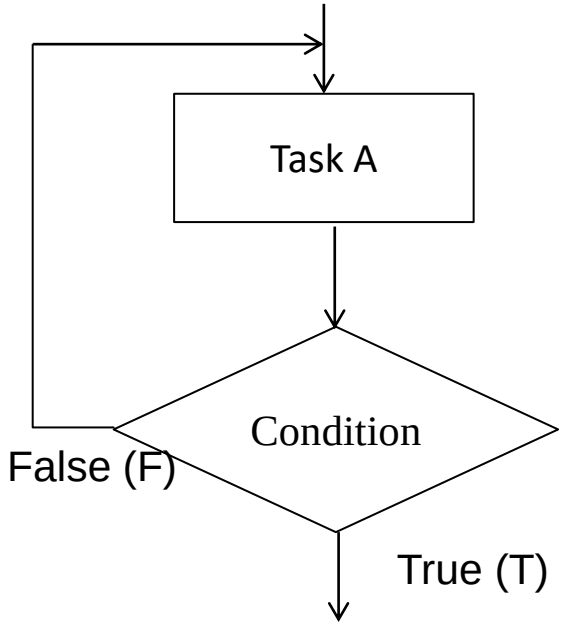
$ax + b = 0$



Loop Structure

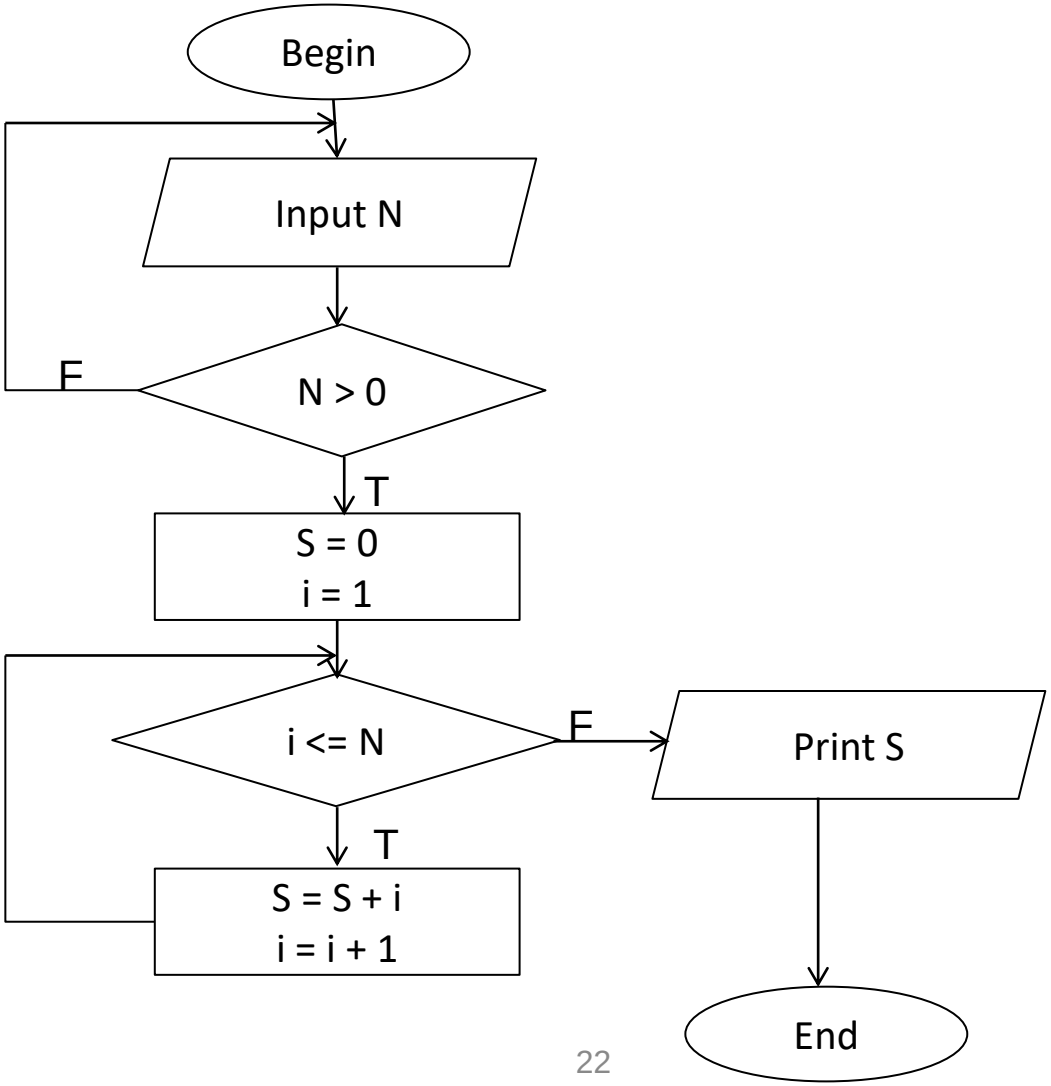
- **Loop** allows a statement or a sequence of statements to be repeatedly executed based on some loop condition





Loop Structure

- Example: Calculate sum S = 1 + 2 + … + N



5. Exercises

Using flowcharts, design algorithms to solve following problems:

1. Find area of Circle.
2. The user will enter the number of eggs gathered and the program will output the number of dozens as well as the number of excess eggs.
3. Check whether given integer value is odd or even.
4. Find Roots of Quadratic equation $ax^2 + bx + c = 0$.
5. Give n is a positive integer, x is a real number. Calculate the sum:

   a) $S(n) = 1+\frac{1}{3}+\frac{1}{5}+...+\frac{1}{2n+1}$

   b) $S(n) = \sum_{i=1}^{n} \frac{x^{2i+1}}{2i+1}\sin ix$

6. Find the factorial of given positive integer N.

Exercises

7. Display all odd numbers between 0 and 1000
8. Check whether given integer value is PRIME or NOT.
9. Find the Fibonacci series till term ≤ 100
10. Given list of numbers a_0, a_1,…., a_{n-1}:
    a) Find sum of all elements in the list. .
    b) Print all elements in the list.
    c) Print all negative elements in the list
    d) Check whether all elements in the list are positive or NOT
    e) Find minimum and maximum element in the list.
    f) Sort the list in increasing order.

Q&A