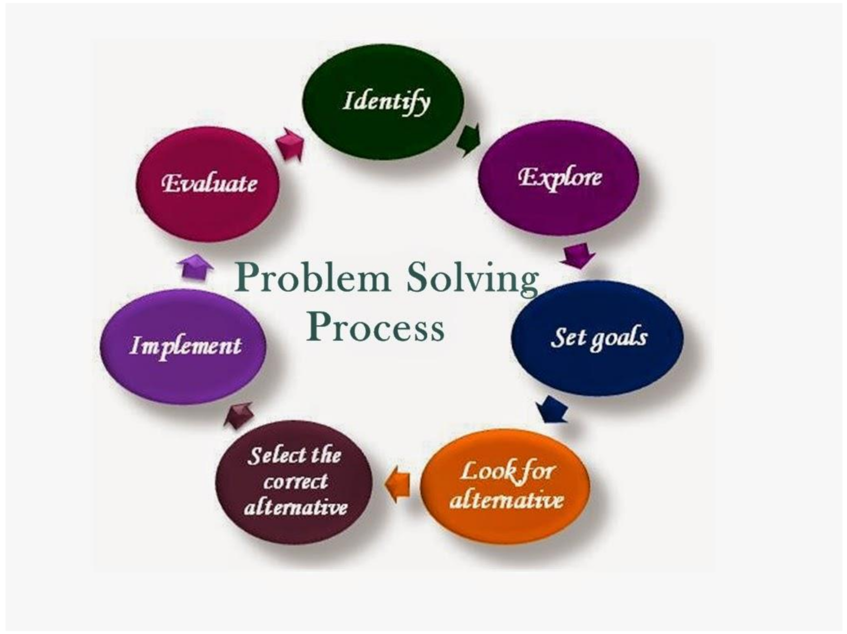
Problem solving

Problem solving is the act of defining a **problem**; determining the cause of the **problem**; identifying, prioritizing, and selecting alternatives for a solution; and implementing a solution.

In programming we focus on implantation of solutions.

Every problem solution starts with a plan. That plan is called an algorithm.



development of an algorithm

The development of an algorithm (a plan) is a key step in solving a problem.

Once we have an algorithm, we can translate it into a computer program in some programming language.

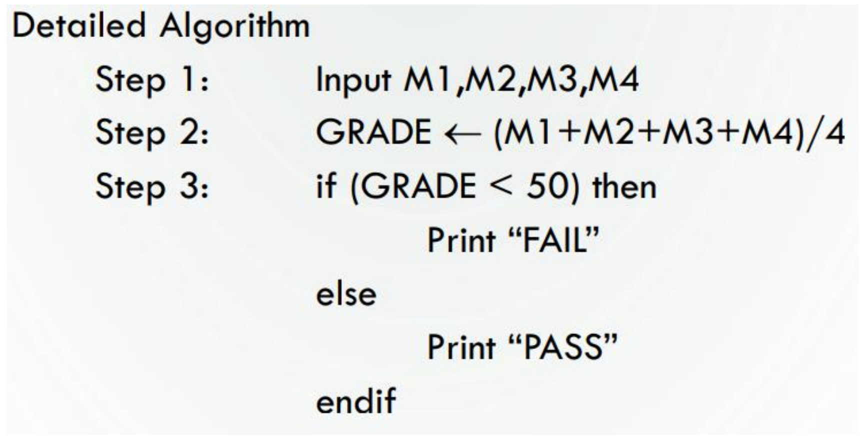
Our algorithm development process consists of five major steps.

Step 1: Obtain a description of the problem.
Step 2: Analyze the problem.
Step 3: Develop a high-level algorithm.
Step 4: Refine the algorithm by adding more detail.
Step 5: Review the algorithm.

Algorithm

An algorithm is used to provide a solution to a particular problem in form of well-defined steps.
Is a set of instruction for solving some problem, step by step .
An algorithm is a plan for solving a problem, but plans come in several levels of detail. It's usually better to start with a high-level algorithm that includes the major part of a solution, but leaves the details until later.

Problem : 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.



Detailed Algorithm

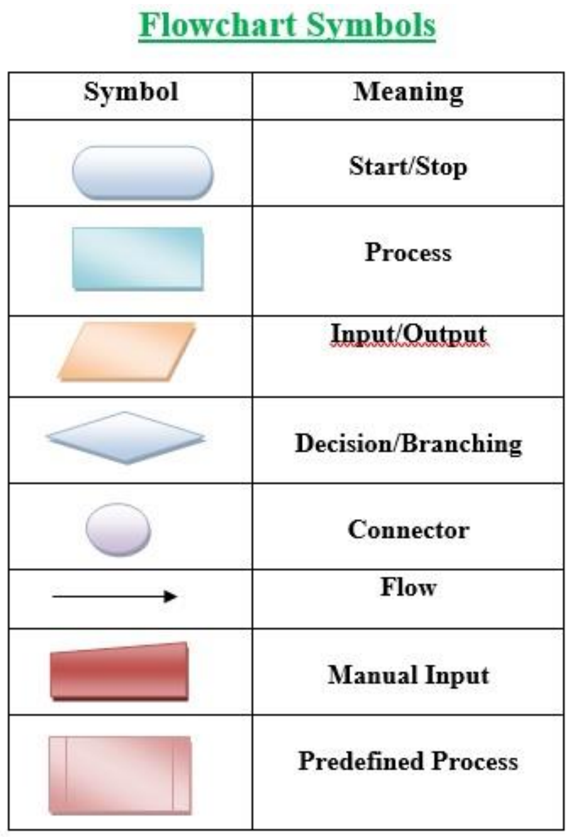
```
Step 1:     Input M1,M2,M3,M4
Step 2:     GRADE ← (M1+M2+M3+M4)/4
Step 3:     if (GRADE < 50) then
                Print "FAIL"
            else
                Print "PASS"
            endif
```

Flowchart

Pseudocode

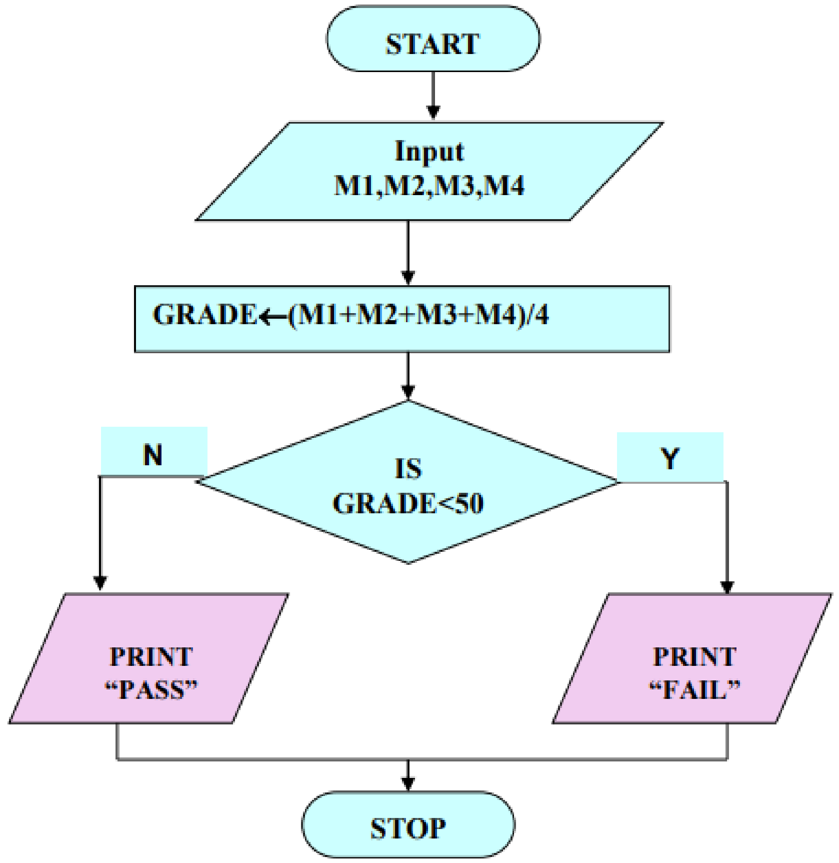
Flowchart

a **flowchart** is **the** pictorial illustration of **the algorithm**. A **flow chart** is **more** understandable as compared to **the algorithm**.



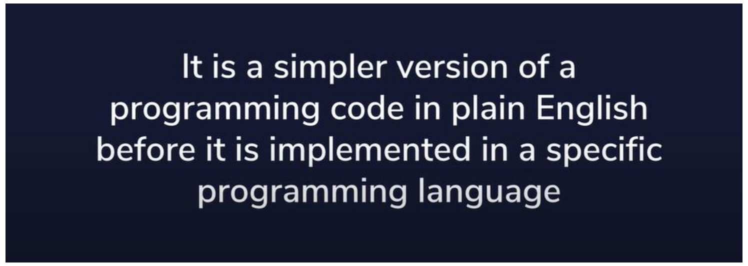
Flowchart Symbols

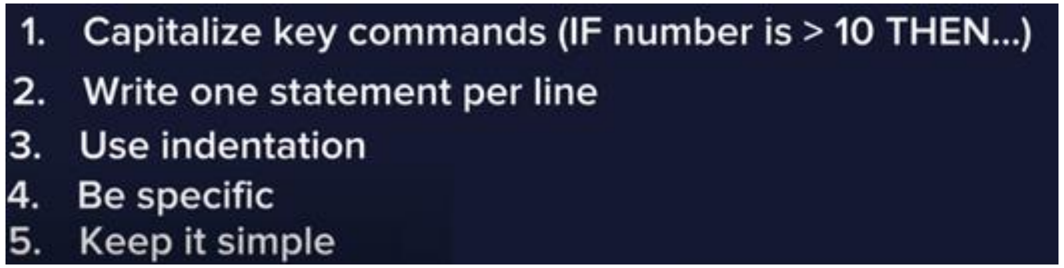
Symbol	Meaning
⬭	Start/Stop
▭	Process
▱	Input/Output
◇	Decision/Branching
○	Connector
→	Flow
	Manual Input
	Predefined Process

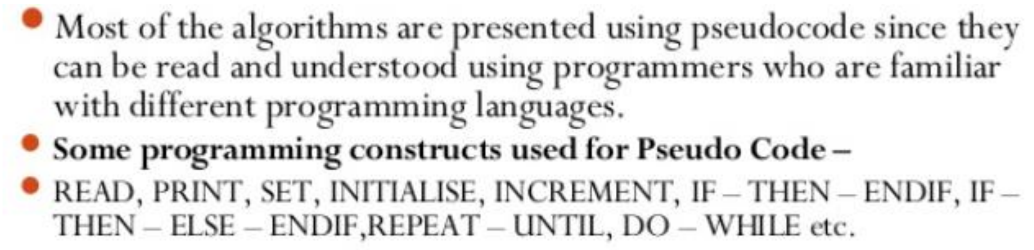


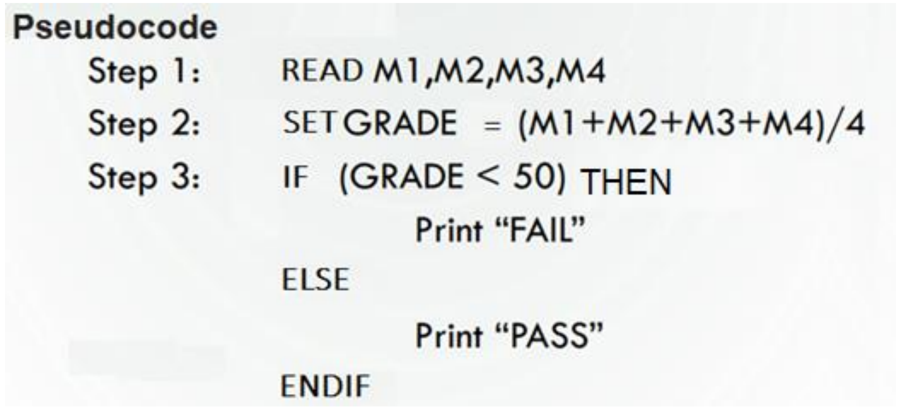
Pseudocode

a **pseudocode** is one of the methods that can be used to represent an **algorithm**.
if algorithm express a bit more formally is Pseudocode .


It is a simpler version of a programming code in plain English before it is implemented in a specific programming language


1. Capitalize key commands (IF number is > 10 THEN...)
2. Write one statement per line
3. Use indentation
4. Be specific
5. Keep it simple


- Most of the algorithms are presented using pseudocode since they can be read and understood using programmers who are familiar with different programming languages.
- **Some programming constructs used for Pseudo Code –**
- READ, PRINT, SET, INITIALISE, INCREMENT, IF – THEN – ENDIF, IF – THEN – ELSE – ENDIF,REPEAT – UNTIL, DO – WHILE etc.


Pseudocode

```
Step 1:    READ M1,M2,M3,M4
Step 2:    SET GRADE  = (M1+M2+M3+M4)/4
Step 3:    IF  (GRADE < 50) THEN
                   Print "FAIL"
           ELSE
                   Print "PASS"
           ENDIF
```

Introduction to Searching Algorithms

Not even a single day pass, when we do not have to search for something in our day to day life, car keys, books, pen, mobile charger.

Linear Search
Binary Search
Jump Search
Interpolation Search
Exponential Search

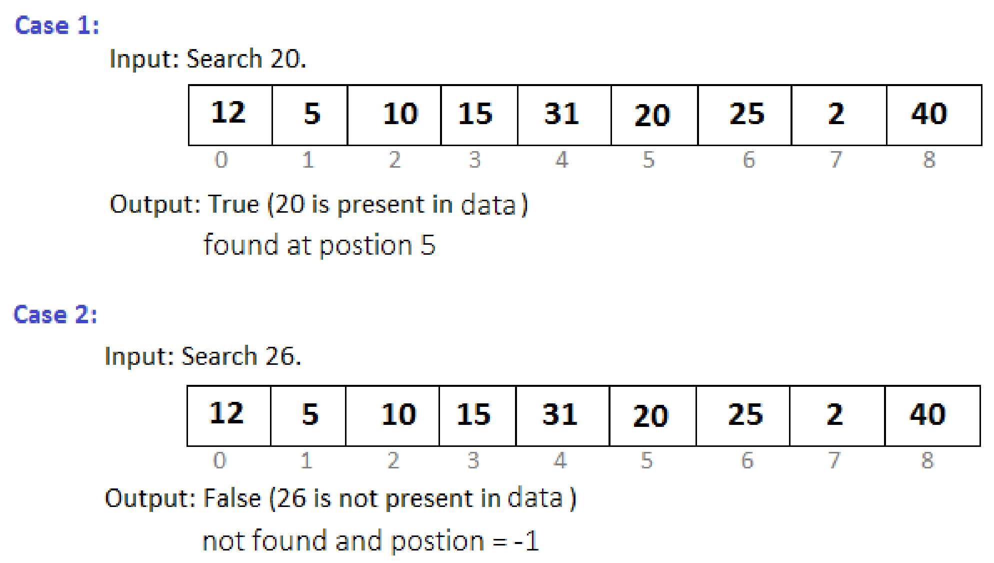
Linear Search

A simple algorithm is to do a linear search:

•Start from the leftmost element of data and one by one compare the target value with each element of data

•If target value matches with an element, return the position or “found”.

•If x doesn’t match with any of elements, return -1 or “not found”.



Case 1:

Input: Search 20.

12	5	10	15	31	20	25	2	40
0	1	2	3	4	5	6	7	8

Output: True (20 is present in data)

found at postion 5

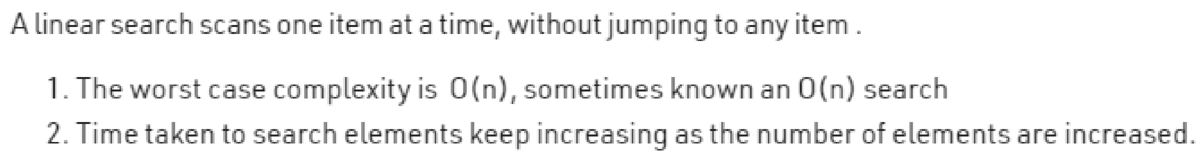
Case 2:

Input: Search 26.

12	5	10	15	31	20	25	2	40
0	1	2	3	4	5	6	7	8

Output: False (26 is not present in data)

not found and postion = -1



A linear search scans one item at a time, without jumping to any item .

1. The worst case complexity is O(n), sometimes known an O(n) search
2. Time taken to search elements keep increasing as the number of elements are increased.

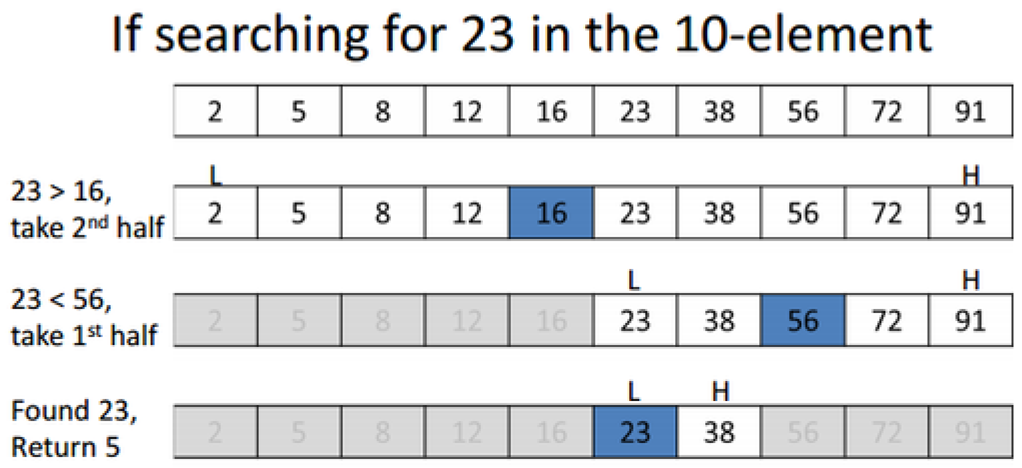
Binary Search

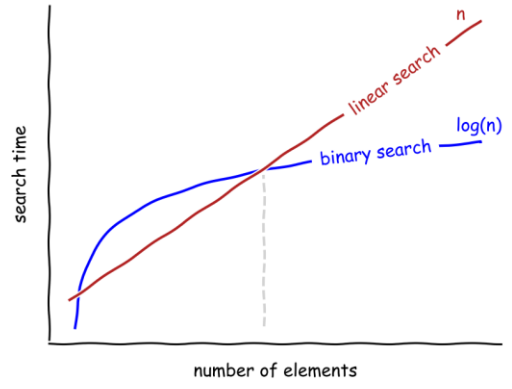
Search a ***sorted data*** by repeatedly dividing the search interval in half.

Begin with an interval covering the whole data.

If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.

Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.


If searching for 23 in the 10-element
2 5 8 12 16 23 38 56 72 91
23 > 16, take 2nd half
L H
2 5 8 12 16 23 38 56 72 91
23 < 56, take 1st half
L H
2 5 8 12 16 23 38 56 72 91
Found 23, Return 5
L H
2 5 8 12 16 23 38 56 72 91


search time
number of elements
linear search
n
binary search
log(n)