Compiling vs. Interpreting

- Compiling is a static (i.e., pre-execution), one-shot translation
  - Once a program is compiled, it may be run over and over again without further need for the compiler or the source code
- Interpreting is dynamic (i.e., happens during execution)
  - The interpreter and the source code are needed every time the program runs
- Compiled programs tend to be faster, while interpreted ones lend themselves to a more flexible programming environments (*they can be developed and run interactively*)

Program Design Techniques

- Pseudocode
- Algorithm
- Flowchart

Design Techniques

- 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

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

Example : Pseudocode

- Write a pseudocode and an algorithm to convert the length in feet to inches

Example : Convert feet into inches

1. Input the length in feet
2. Calculate the length in inches by multiplying length in feet with 12
3. Print length in inches

Example : Algorithm

- Step 1: Input L_ft
- Step 2: L_inches $\longleftarrow$ L_ft x 12
  Step 3: Print L_inches

Flowchart

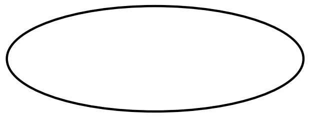
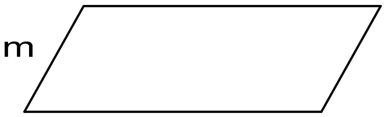
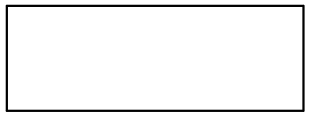
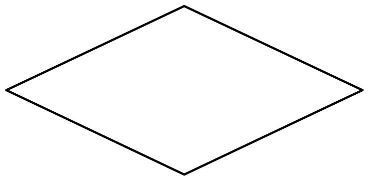
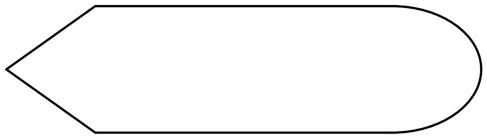
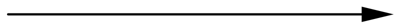
- Flowchart is schematic representation of a sequence of operations, as in a manufacturing process or computer program.
- It is a graphic representation of how a process works, showing, at a minimum, the sequence of steps.
- A flowchart consists of a sequence of instructions linked together by arrows to show the order in which the instructions must be carried out

The Flowchart

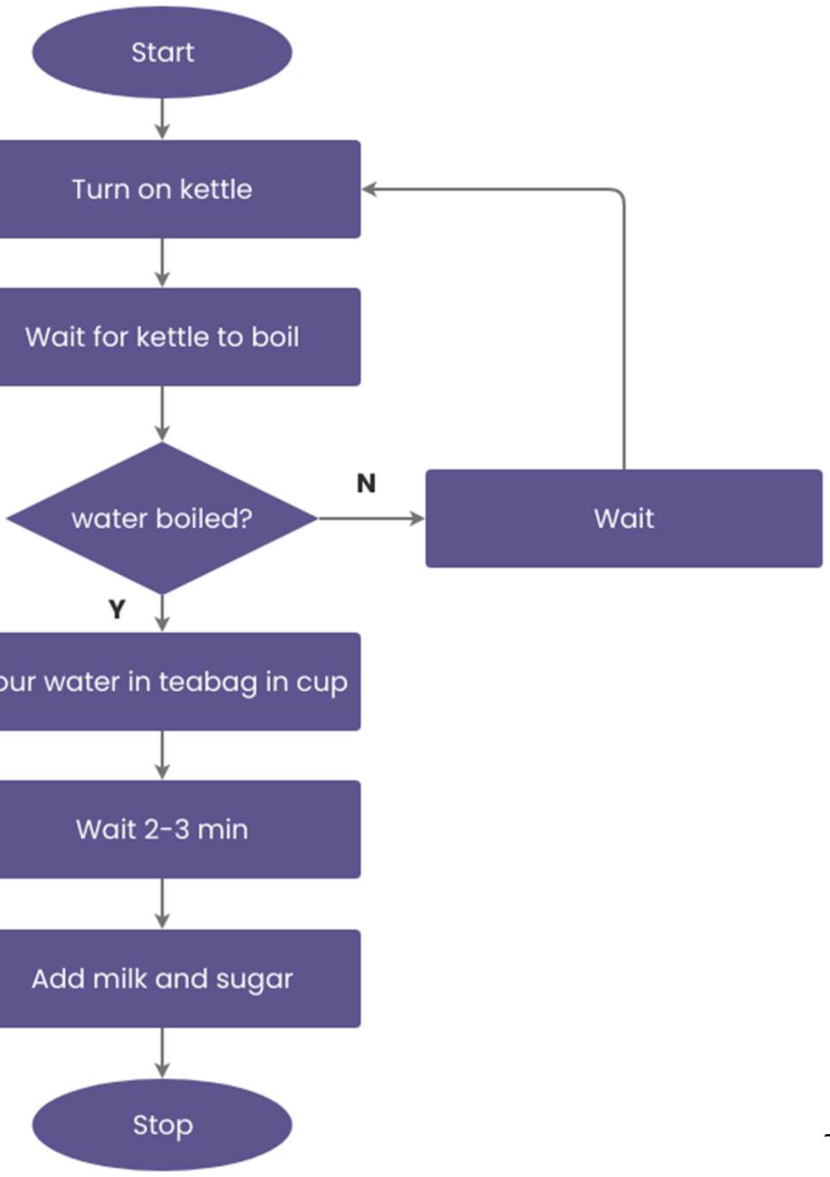
A Flowchart

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

Flowchart Symbols

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

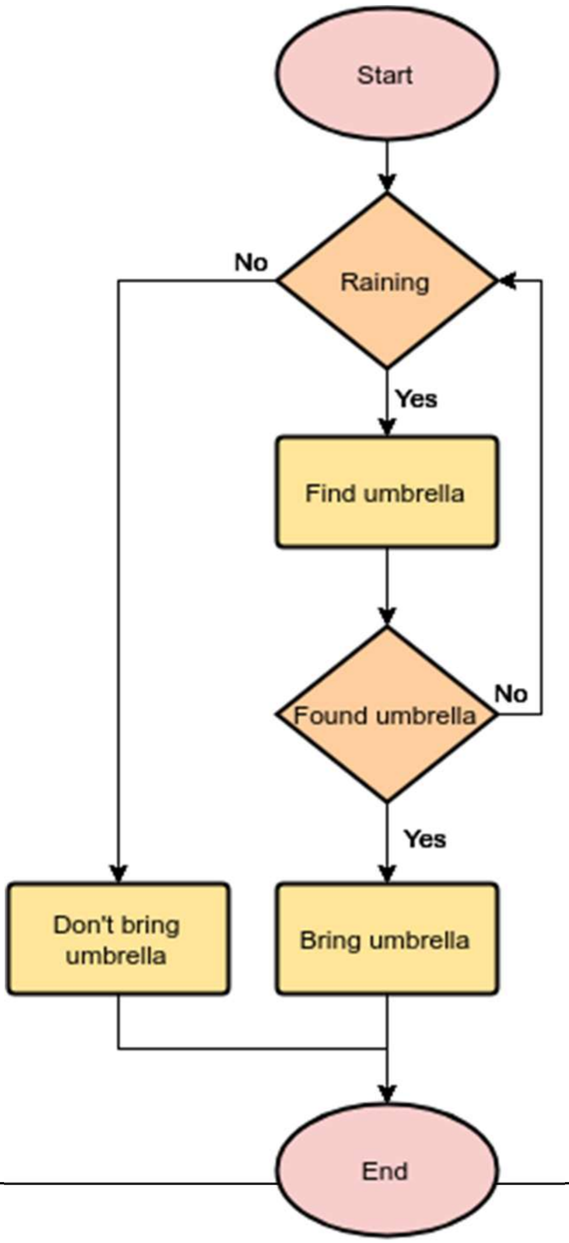
Making a Cup of Tea



Flowchart Example

- Should I bring an umbrella to university ?
  - Process through which a person decides whether to bring an umbrella to university or not

Should I bring umbrella ?



Flow Chart Exercise

- Completing and submitting an assignment ?

Pseudocode & Algorithm

- **Excercise:** 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:

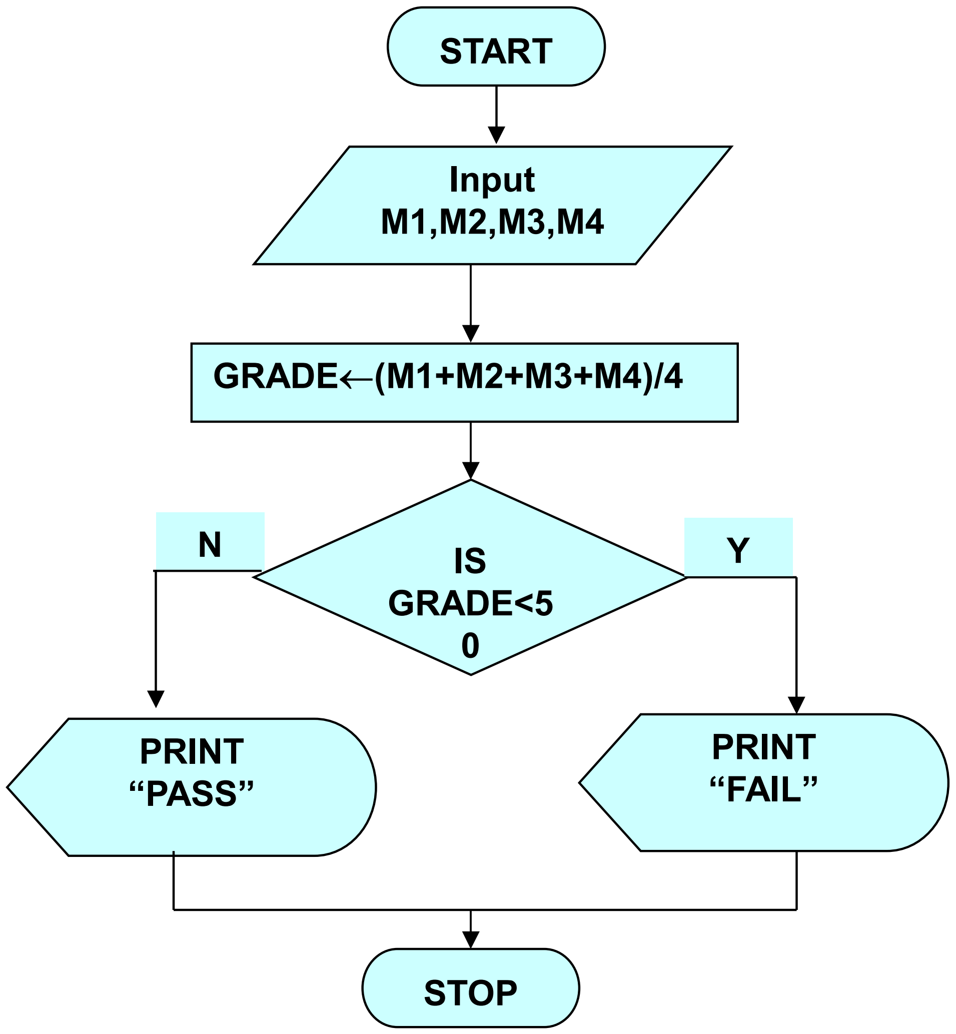
- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

```
if average is below 50
        Print “FAIL”
else
        Print “PASS”
```

Pseudocode & Algorithm

- 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
```

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
```