ALGORITHMS AND FLOWCHARTS

- 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

- First produce a general algorithm (one can use **pseudocode**)
- Refine the algorithm successively to get step by step detailed **algorithm** that is very close to a computer language.
- **Pseudocode** is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.
Example 1: 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
  - Print “FAIL”
else
  - Print “PASS”
Pseudocode & Algorithm

- Detailed Algorithm
- **Step 1:** Input \(M_1, M_2, M_3, M_4\)
- **Step 2:** \(\text{GRADE} \leftarrow (M_1 + M_2 + M_3 + M_4)/4\)
- **Step 3:** if \((\text{GRADE} < 50)\) then
  
  Print “FAIL”

else

  Print “PASS”

endif
The Flowchart

- (Dictionary) A schematic representation of a sequence of operations, as in a manufacturing process or computer program.
- (Technical) A graphical representation of the sequence of operations in an information system or program. Information system flowcharts show how data flows from source documents through the computer to final distribution to users. Program flowcharts show the sequence of instructions in a single program or subroutine. Different symbols are used to draw each type of flowchart.
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

### Basic

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Use in Flowchart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oval</td>
<td><img src="image" alt="Oval" /></td>
<td>Denotes the beginning or end of the program</td>
</tr>
<tr>
<td>Parallelogram</td>
<td><img src="image" alt="Parallelogram" /></td>
<td>Denotes an input operation</td>
</tr>
<tr>
<td>Rectangle</td>
<td><img src="image" alt="Rectangle" /></td>
<td>Denotes a process to be carried out e.g. addition, subtraction, division etc.</td>
</tr>
<tr>
<td>Diamond</td>
<td><img src="image" alt="Diamond" /></td>
<td>Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)</td>
</tr>
<tr>
<td>Hybrid</td>
<td><img src="image" alt="Hybrid" /></td>
<td>Denotes an output operation</td>
</tr>
<tr>
<td>Flow line</td>
<td><img src="image" alt="Flow line" /></td>
<td>Denotes the direction of logic flow in the program</td>
</tr>
</tbody>
</table>
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

START

Input M1, M2, M3, M4

GRADE ← (M1 + M2 + M3 + M4) / 4

IS GRADE < 50

N

PRINT “PASS”

Y

PRINT “FAIL”

STOP
Example 2

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

**Pseudocode:**

- *Input the length in feet (Lft)*
- *Calculate the length in cm (Lcm) by multiplying LFT with 30*
- *Print length in cm (LCM)*
Example 2

Algorithm

- Step 1: Input Lft
- Step 2: $\text{Lcm} \leftarrow \text{Lft} \times 30$
- Step 3: Print Lcm
Example 3

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

Pseudocode

- Input the width (W) and Length (L) of a rectangle
- Calculate the area (A) by multiplying L with W
- Print A
Example 3

Algorithm

- Step 1: Input $W,L$
- Step 2: $A \leftarrow L \times W$
- Step 3: Print $A$
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: 
  \( x_1 = \frac{-b + d}{2a} \) and \( x_2 = \frac{-b - d}{2a} \)
Example 4

Pseudocode:

- *Input the coefficients* *(a, b, c)* *of the quadratic equation*
- *Calculate* $d$
- *Calculate* $x_1$
- *Calculate* $x_2$
- *Print* $x_1$ *and* $x_2$
Example 4

**Algorithm:**

- **Step 1:** Input \( a, b, c \)
- **Step 2:** \( d \leftarrow \sqrt{b \times b - 4 \times a \times c} \)
- **Step 3:** \( x_1 \leftarrow (-b + d) / (2 \times a) \)
- **Step 4:** \( x_2 \leftarrow (-b - d) / (2 \times a) \)
- **Step 5:** Print \( x_1, x_2 \)
DECISION STRUCTURES

- The expression $A > B$ is a logical expression
- *it describes a condition we want to test*
- *if $A > B$ is true (if $A$ is greater than $B$) we take the action on left*
- print the value of $A$
- *if $A > B$ is false (if $A$ is not greater than $B$) we take the action on right*
- print the value of $B$
DECISION STRUCTURES

- If A is greater than or equal to B, print B.
- If A is less than B, print A.
IF–THEN–ELSE STRUCTURE

The structure is as follows

If condition then
   true alternative
else
   false alternative
endif
The algorithm for the flowchart is as follows:

If $A > B$ then

print $A$

else

print $B$

endif
# Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>≥</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>≤</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>≠</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
Example 6

Write an algorithm that reads two values, determines the largest value and prints the largest value with an identifying message.

**ALGORITHM**

Step 1: \( \text{Input VALUE1, VALUE2} \)

Step 2: \( \text{if (VALUE1 > VALUE2) then} \)

\[ \text{MAX } \leftarrow \text{VALUE1} \]

\( \text{else} \)

\[ \text{MAX } \leftarrow \text{VALUE2} \]

\( \text{endif} \)

Step 3: \( \text{Print “The largest value is”, MAX} \)
Example 6

START

Input
VALUE1, VALUE2

Y

is
VALUE1 > VALUE2

N

MAX ← VALUE1

MAX ← VALUE2

Print
"The largest value is", MAX

STOP
LOOPS

- Computers are particularly well suited to applications in which operations are repeated many times.
- If the same task is repeated over and over again a loop can be used to reduce program size and complexity.
Example 7: Write an algorithm and draw a flowchart to calculate $2^4$.

- **Algorithm:**
  - Step 1: Base $\leftarrow 2$
  - Step 2: Product $\leftarrow$ Base
  - Step 3: Product $\leftarrow$ Product $\ast$ Base
  - Step 4: Product $\leftarrow$ Product $\ast$ Base
  - Step 5: Product $\leftarrow$ Product $\ast$ Base
  - Step 6: *Print* Product
Flowchart

START

Base ← 2

Product ← Base

Product ← Product * Base

Product ← Product * Base

Product ← Product * Base

Print Product

STOP
Question: What happens if you want to calculate 2 to the power of 1000?

Answer: Use a LOOP (repeated execution of the same set of instructions)
Example 8:

- Write an algorithm and draw a flowchart to calculate $2^4$ using a loop approach? Verify your result by a trace table.
Algorithm:

Step 1: Base ← 2
Step 2: Power ← 4
Step 3: Product ← Base
Step 4: Counter ← 1
Step 5: While Counter < Power
  Repeat Step 5 through step 7
Step 6: Product ← Product * Base
Step 7: Counter ← Counter + 1
Step 8: Print Product
START

Base ← 2
Power ← 4

Product ← Base
Counter ← 1

is
Counter < Power

N

Y

Product ← Product * Base

Counter ← Counter + 1

Print Product

STOP
## TRACING

<table>
<thead>
<tr>
<th>Base</th>
<th>Power</th>
<th>Product</th>
<th>Counter</th>
<th>Counter &lt; Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 1:</td>
<td>2</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>STEP 2:</td>
<td>2</td>
<td>4</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>STEP 3:</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>STEP 4:</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>STEP 5:</strong></td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>STEP 6:</td>
<td>2</td>
<td>4</td>
<td>2x2=4</td>
<td>1</td>
</tr>
<tr>
<td>STEP 7:</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>1+1=2</td>
</tr>
<tr>
<td><strong>STEP 5:</strong></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>STEP 6:</td>
<td>2</td>
<td>4</td>
<td>4x2=8</td>
<td>2</td>
</tr>
<tr>
<td>STEP 7:</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>2+1=3</td>
</tr>
<tr>
<td><strong>STEP 5:</strong></td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>STEP 6:</td>
<td>2</td>
<td>4</td>
<td>8x2=16</td>
<td>3</td>
</tr>
<tr>
<td>STEP 7:</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>3+1=4</td>
</tr>
<tr>
<td><strong>STEP 5:</strong></td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
</tr>
</tbody>
</table>

**Step 8:** print 16.

**Step 1:** Base ← 2
**Step 2:** Power ← 4
**Step 3:** Product ← Base
**Step 4:** Counter ← 1
**Step 5:** While Counter < Power
  Repeat Step 5 through step 7
**Step 6:** Product ← Product * Base
**Step 7:** Counter ← Counter +1
**Step 8:** *Print* Product
Example 10: Write down an algorithm and draw a flowchart to find and print the largest of three numbers. Read numbers one by one. Verify your result by a trace table. (Use 5, 7, 3 as the numbers read)
Algorithm

- Step 1: \textit{Input} N1
- Step 2: Max $\leftarrow$ N1
- Step 3: \textit{Input} N2
- Step 4: \textit{If} (N2$>$Max) \textit{then}
  \hspace{1cm} Max = N2
\hspace{1cm} \textit{end if}
- Step 5: \textit{Input} N3
- Step 6: \textit{If} (N3$>$Max) \textit{then}
  \hspace{1cm} Max = N3
\hspace{1cm} \textit{end if}
- Step 7: \textit{Print} “The largest number is:”, Max
Flowchart & Tracing

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>Max</th>
<th>N2&gt;Max</th>
<th>N3&gt;Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: 5</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Step 2: 5</td>
<td>?</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Step 3: 5</td>
<td>7</td>
<td>?</td>
<td>5</td>
<td>T</td>
<td>?</td>
</tr>
<tr>
<td>Step 4: 5</td>
<td>7</td>
<td>?</td>
<td>7</td>
<td>T</td>
<td>?</td>
</tr>
<tr>
<td>Step 5: 5</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Step 6: 5</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Step 8: Print → Largest Number is 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

START

INPUT N1

MAX—N1

INPUT N2

N2>MAX

Y

MAX—N2

INPUT N3

N3>MAX

Y

PRINT "Largest Number is", MAX

STOP
Example 11: Write down an algorithm and draw a flowchart to find and print the largest of N (N can be any number) numbers. Read numbers one by one. Verify your result by a trace table. (Assume N to be 5 and the following set to be the numbers \{1 4 2 6 8 \})
Algorithm:

- **Step 1:** Input $N$
- **Step 2:** Input $X$
- **Step 3:** $\text{Max} \leftarrow \text{Current}$
- **Step 4:** $\text{Counter} \leftarrow 1$
- **Step 5:** \textbf{While} (Counter < $N$)
  - Repeat steps 5 through 8
- **Step 6:** $\text{Counter} \leftarrow \text{Counter} + 1$
- **Step 7:** Input $X$
- **Step 8:** \textbf{If} ($X > \text{Max}$) then
  - $\text{Max} \leftarrow X$
\textbf{endif}
- **Step 9:** \textbf{Print} Max
How many times will steps 4, 6, and 7 be executed?
Do Loops

- It is convenient to introduce a special type of loop that is headed by a special macroinstructions.

- This terminology comes from FORTRAN, although many programming languages have this type of loop.
For example:

**BASIC**

DO K=1 to N
{body of loop}
END;

**FORTRAN**

Do n K=1 , N
{body of loop}
n CONTINUE
Example: A company has 80 employees give a flowchart that
finds the average salary and the number of employees earning above the average salary. Observe that the salaries are read into an array, SALARY. Next, the average salary, AVG, is calculated.

Then each salary, SALARY(K), is compared with AVG to obtain the number NUM of salaries greater than AVG.
Start

Dimension SALARY (80)

Read SALARY

SUM=0

Do K=1 to 80

SUM=SUM+SALARY(K)

AVG=SUM/80

NUM=0

Do K=1 to 80

SALARY>AVG

Yes

NUM=NUM+1

No

(a)

No

Print SALARY, AVG, NUM

Stop
PROBLEMS
Prob. 1. Write an algorithm and draw a flowchart to print the square of all numbers from 1 to 10.

Prob. 2. Write an algorithm and draw a flowchart to print the SUM of numbers from LOW to HIGH. Test with LOW=3 and HIGH=9.

Prob. 3. Write an algorithm and draw a flowchart to print all numbers between LOW and HIGH that are divisible by NUMBER.

Prob. 4. Draw a flowchart for a program that reads 10 numbers from the user and prints out their sum, and their product.
Prob. 5. Write an algorithm and draw a flowchart to count and print all numbers from LOW to HIGH by steps of STEP. Test with LOW=0 and HIGH=100 and STEP=5.

Prob. 6. Write an algorithm and draw a flowchart to print the multiplication table for 6's. i.e.

---- 1 × 6 = 6
---- 2 × 6 = 12
...
---- 12 × 6 = 72

Prob. 7. Write an algorithm and draw a flowchart that will find and print the product of 3 numbers.
Prob. 8. Write an algorithm and draw a flowchart that will find and print
The factorial of NUMBER is FACTORIAL.
Test the flowchart for NUMBER=5.