PRACTICE SHEET

What You Will Learn:

Flowcharts & Pseudocode Design, Decision Trees, Basic Scratch Programming, Introduction to Loops(via Flowcharts), Introduction to Graphs, Introduction to Matrices and Logic Gates, Introduction to Algorithms and Time Complexity

---

[MEDIUM] Q1) A robot is programmed to pass through a door using the following control logic:

"If the door is closed, then the robot first checks if the door is locked. If the door is locked, it unlocks the door. Then, it opens the door and moves forward."

The simplified pseudocode is:

```
if door is closed:
  if door is locked:
    unlock door()
  open door()
advance()
```

Task:
Design a **flowchart** that represents this logic clearly. Your flowchart must include:

- Decision blocks for checking if the door is closed or locked
- Action blocks for unlocking, opening, and advancing
- Proper start and stop symbols
- Flow lines with appropriate condition labels (Yes/No)

Note: This problem tests your ability to convert conditional logic into a correct and readable flowchart format.

Hint: Use decision diamonds and process rectangles to represent conditions and actions respectively.

[EASY] Q2) Create a Scratch game where:

- A sprite moves 10 steps when the spacebar is pressed
- It says "Ouch!" if it touches the edge

[EASY] Q3) Imagine it’s 8 PM and you’re hungry. You decide to make Maggi.” You don’t randomly throw things in a pan.
You follow a **sequence of steps**:

1. Boil 1 cup of water
2. Add noodles
3. Wait 2 minutes (Is 2 minutes enough?)
4. Add tastemaker
5. Stir
6. Wait 1 more minute
7. Serve hot

Conclusion: This is an **algorithm** - a **step-by-step procedure** to solve a problem (in this case, hunger!).

Definition: What is an Algorithm?

An **algorithm** is a **finite sequence of well-defined steps** to solve a problem or perform a task.

- It has a **start**
- It must **end**
- Each step should be **clear**
- It should **solve the problem**

A Technical Example:

Task: Find the largest number among three numbers.

Let’s say:

a = 20

b = 45

c = 15

Algorithm Steps:

1. Compare a and b
2. Store the larger in max
3. Compare max with c
4. Update max if c is larger
5. Print max

That's it! This can now be written in any programming language.

Task: Write an algorithm to **find if a number is even or odd.**

Introducing Time Complexity – The Speed of an Algorithm
Two friends race to make Maggi.

- One follows the exact 7 steps.
- The other takes unnecessary detours — adds cheese, fries veggies, etc.

Who finishes first? The one with **fewer, simpler steps**.

Similarly, in computers, we **compare algorithms** based on **how fast they run**.

Time complexity tells us **how the time taken by an algorithm grows** as the input size increases.

[MEDIUM] Q4) A robot must clean a room:

```
If room is dirty:
   start vacuum
   clean for 10 minutes
   stop vacuum
Else:
   print "Already clean"
```

Tasks: Draw a flowchart

[MEDIUM] Q5) **Smart Power Distribution Panel**

Problem Statement:
A smart power panel allows supply to critical units **only if**:

- The backup battery is charged **AND**
- Either solar power is ON **OR** main supply is ON
  AND
- The load is under the max threshold

Draw a flowchart and write a pseudocode for this question.

[MEDIUM] Q6) Problem:
You need to stir a tank **until the mixture reaches pH 7**.
Every time you stir, check the pH.
If pH is dangerous (e.g., <3 or >11), **stop immediately** (safety break).
If pH is between 4 and 6 or 8 and 10, **skip that reading** and wait for the next (continue).

Write pseudocode for the described problem.

[HARD] Q7) In a factory setting, the goal is to keep track of the number of good products passing through a conveyor belt. The factory processes products continuously, but only good products should be counted, while defective ones must be skipped.

The process starts with a counter set to zero and a variable to keep track of the number of products processed. As products move along the conveyor, the factory checks each one to determine whether it is defective or not. The factory will continue processing products until it reaches 100 processed items.

If a product is found to be defective, it is immediately skipped, and a message indicating that the product was defective is printed. The counter for good products remains unchanged for that iteration, ensuring only the non-defective products are counted. On the other hand, if the product is good, the counter is incremented, adding to the total number of good products.

After processing 100 products, the factory will print the total number of good products that passed the quality check. This ensures an efficient tracking of good items, while minimizing the impact of defective ones.

Write pseudocode for the described problem.

[MEDIUM] Q8) A young adventurer named Luffy is lost in a mysterious forest. To find his way out, he encounters several forks in the road. At each fork, Luffy needs to make a decision. Based on his choices, he will either find his way to safety or get stuck deeper into the forest.

Here are the decision points Luffy encounters:

- **First Fork**: Luffy comes to a fork in the road. He can either go **Left** or **Right**.
    - If he goes **Left**, he encounters a river and must decide if he wants to **Swim** across or go **Back**.
    - If he goes **Right**, he meets a cave and must decide whether to **Enter** or go **Back**.
- **River Decision**:
    - If Luffy chooses to **Swim**, he may either encounter a **Friendly Dragon** who helps him escape, or a **Snake** that tries to stop him. The Dragon will guide Luffy to safety, while the Snake will send him back to the first fork.
    - If Luffy chooses to **Backtrack**, he returns to the first fork.
- **Cave Decision**:
    - If Luffy chooses to **Enter**, he must navigate a maze inside. The maze has two possible exits: one leads to safety, and the other to a **Dark Pit** where he gets stuck.
    - If Luffy chooses to **Backtrack**, he returns to the first fork.

Task:

- Draw a decision tree that represents Luffy's choices at each fork in the road.
- For each path, show the possible outcomes (success or failure).

Purpose: This activity encourages students to break down a scenario into logical steps, visualize decision points, and understand the structure of algorithms.

[EASY] Q9) Draw a flowchart to check if a given number is **positive or negative**. Include decision blocks for the condition and different paths for "True" and "False".

[EASY] Q10) Draw a flowchart to print numbers from **1 to 10** using a **while loop**. Show the initialization, condition check, and update steps clearly.

[EASY] Q11) Draw a flowchart to print the first **5 even numbers (2, 4, 6, 8, 10)** using a **for loop**. Include loop initialization, condition, increment, and output steps.

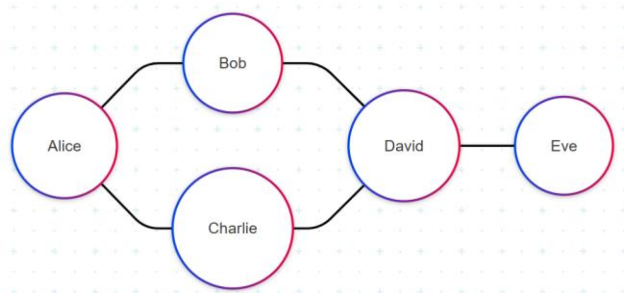
[EASY] Q12) Draw a flowchart to take user input repeatedly until the user enters **'0' (zero)**. Ensure the loop executes **at least once** before checking the condition.

[MEDIUM] Q13) You're building a feature for a social media app to find mutual friends between users. How would you represent the relationships (who is friends with whom) so that the computer can efficiently solve this?

Users = **Vertices (Nodes)**

Friendships = **Edges** (lines connecting nodes).

Undirected Graph (since friendship is mutual).



Interpretation:

- Alice is friends with Bob and Charlie.
- Bob is friends with David.
- David is also friends with Eve and Charlie.

a) How would you find all mutual friends between Alice and David?

b) Is there a path connecting Eve to Alice? If yes, what is it?

c) What if some friendships are one-directional (e.g., Instagram follows)? How would the graph change?

[EASY] Q14) **1. Introduction to Matrices in Computing**

In mathematics, you have worked with matrices as rectangular arrangements of numbers in rows and columns. In computers, matrices are used to represent structured data, such as images. Consider the following 3x3 matrix where each number represents a pixel intensity (0 = black, 1 = white)

[[0, 1, 0] ,

[1, 0, 1] ,

[0, 1, 0]]

Draw the corresponding 3x3 black-and-white image by shading the squares (0 = filled, 1 = empty). What shape do you see?

2. How Computers Store Images

Digital images are made of tiny dots called **pixels** (picture elements). Each pixel has a numerical value representing its color or brightness.

In grayscale images:

- **0** = Pure black
- **255** = Pure white
- Values in between represent shades of gray.

[EASY] Q15) Computers process everything (numbers, text, images) as **binary (0s and 1s)** using tiny electronic switches called **logic gates**.

Real-World Analogy - Light Switch:

ON (1) = Electricity flows.

OFF (0) = No electricity.

Basic Gates (With Truth Tables):

Gate	Symbol	Behavior
AND	A & B → Output	Output=1 **only if** both inputs=1
OR	A | B → Output	Output=1 **if any** input=1
NOT	A → Output	Flips input (1→0, 0→1)

1. How Computers Use Logic Gates

Example: Adding two numbers (1 + 1 = 10 in binary) uses a **circuit of gates** called a *half-adder*:

1. **XOR Gate** calculates the sum bit (1 XOR 1 = 0).
2. **AND Gate** calculates the carry bit (1 AND 1 = 1).

Problem:
You're designing a security system that unlocks a door ONLY if:

- **Condition 1:** A valid key card is swiped (**Input A**).
- **Condition 2:** A 4-digit PIN is entered correctly (**Input B**).

a. Which logic gate (AND/OR/NOT) should connect Input A and B? Why?

b. Draw the truth table for this system:

A (Key Card)	B (PIN)	Door Unlocks?
0 (No)	0	?
0	1	?
1	0	?
1	1	?