# PRACTICE SHEET

**What You Will Learn:**

IP Addressing Basics, Cloud Computing Introduction, Flowchart Design, Deadlock, Pseudocode Structure, Operating System Introduction, Recursion Basics, Flowgorithm

---

[EASY] Q1) IP Address - Your Device's Digital Address

Imagine you want to send a Rakhi to your brother living far away. You can't just throw the package in the air - it needs a clear destination with a pin code, street name, and house number.

Similarly, when your computer wants to send a message across the internet - like opening a website or sending a file - it must know where to send it. That "where" is called the IP Address - your device's unique digital home address.

Just as every house needs a unique address, every internet-connected device gets a unique IP address so messages reach the correct place.

**Definition: What is an IP Address?**

An IP Address (Internet Protocol Address) is a unique numerical label assigned to every device connected to a computer network that uses the Internet Protocol for communication.

It serves two main functions:

1. Identification – Who is sending/receiving the data?
2. Location Addressing – Where is the data going?

**Task - What's Your IP Address?**

**Option A (Offline Method):**

Open the Command Prompt or Terminal and type:

ipconfig (on Windows)

ifconfig (on Mac/Linux)

Look for your IPv4 Address (e.g., 192.168.1.5)

**Option B (Online Tool):**

Visit https://whatismyipaddress.com or search "What is my IP" on Google.

---

[MEDIUM] Q2) Imagine two robots, R1 and R2, working together to paint a long wall.

- R1 picks up the brush first.
- R2 picks up the paint bucket first.

Now, each robot waits for the other item to become available so they can paint.
But since neither will let go of what they're holding… they're stuck forever.

This is called a **Deadlock** when two (or more) processes **wait on each other** forever and nothing moves forward.

**Key Concepts:**

| **Term** | **Meaning** |
|---|---|
| **Concurrency** | When two or more tasks **run at the same time**, possibly interacting. |
| **Deadlock** | When two or more tasks **wait on each other's resources**, and none can continue. |

**Simulation of a Deadlock (Pseudocode)**

Robot R1:

  Acquire Brush

  Wait for Paint Bucket

  Paint wall

Robot R2:

  Acquire Paint Bucket

  Wait for Brush

  Paint wall

// Both robots are now waiting for each other forever...

END

**Note:** Deadlock happens when multiple processes lock some resources and wait for others, creating a cycle of dependency.

**Task:** Suggest a fix using a timeout and write its pseudocode.

---

[EASY] Q3) Imagine this:
You typed your notes on your laptop using Google Docs. Later that evening, you open your phone, log in, and your notes are right there!

But how? You didn't email them to yourself or use a pen drive.

That's the Cloud - your invisible digital backpack that's always with you wherever you go as long as you have internet.

Just like Doraemon's magic pocket, the Cloud stores your stuff safely in a remote place and lets you access it anytime, from any device.

**Definition: What is Cloud Computing?**

**Cloud computing** means storing and accessing data or programs over the internet instead of your local hard drive.

You don't carry physical storage anymore, your files live on someone else's secure computer (called a **server**), and you get access through the internet.

**Tasks:**

1. Give 3 examples of cloud-based apps you use.
2. Explain: What happens if you lose your device? Will you still have your files?
3. What command or UI lets you sync a folder with a cloud service like Google Drive or GitHub?
   **Hint:** GitHub is not cloud storage, but version-controlled code hosting in the cloud.

---

[EASY] Q4) A man must cross a river with a fox, a goose, and a bag of grain.
His boat can only carry him and one item at a time.

**Rules:**

- If left alone:
    - The fox will eat the goose

- The goose will eat the grain

**Task:**

- Draw a flowchart or state diagram to decide the correct sequence of crossings.
- Use decision diamonds to check “Is anything being eaten?”
- Use state labels like (Left Bank: F, G, B | Boat: empty | Right Bank: -)

Goal: Get all items to the other side safely.

---

[EASY] Q5) You’re standing next to a river with:

- A 5-liter jug
- A 3-liter jug
- An unlimited supply of water

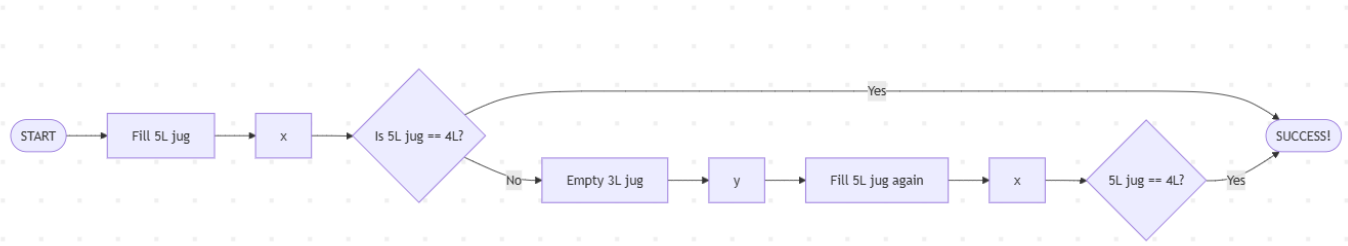
There are no measuring marks, and you cannot estimate.

**Mission:** Get exactly 4 liters of water in one of the jugs.

**Challenge:** You must use a precise sequence of actions like pouring, filling, emptying and reach the goal with logic, not luck.

**Allowed Actions:**

- Fill either jug fully from the river
- Empty either jug
- Pour water from one jug to another, until:
  - The first jug is empty, or
  - The second jug is full



Find x, y and z.

---

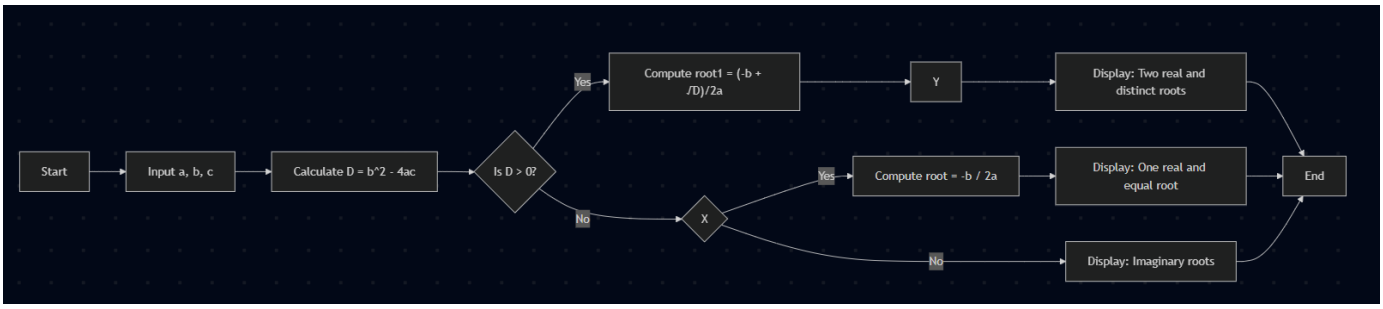
[HARD] Q6) Write an algorithm(pseudocode) and draw a flowchart using flowgorithm to calculate the value of 2 raised to the power 4 using **only a loop** (no built-in power operator). Your algorithm must use only:

- A single loop
- Multiplication
- Integer variables

Then, display the final result.

---

[EASY] Q7) Draw a flowchart to find all the roots of a quadratic equation $ax^2+bx+c = 0$.




Find X and Y and try to make it in flowgorithm.

---

[MEDIUM] Q8) Imagine your classroom is a computer.

Now meet the class monitor, they:

- Decide who speaks next (like managing the CPU)
- Make sure there's space on the board and benches (like memory)
- Keep the notebooks safe and hand them out when needed (like files)
- Stop 5 people from shouting at once (like process control)
- Help students use lab equipment one at a time (like hardware management)

Just like this monitor, the Operating System (OS) is the software that manages and coordinates everything inside your computer silently, smartly, and constantly.

**Definition:**

An **Operating System (OS)** is a system software that acts as a manager between hardware and user applications.
It handles CPU, memory, files, devices, and security like a boss behind the scenes.

**How an OS loads and runs a program-**

1. Start
2. User clicks a program
3. OS finds it in storage
4. Loads it into memory
5. Assigns it to the CPU
6. Shows output
7. End

Draw this as a flowchart (or using Flowgorithm if online).

---

[MEDIUM] Q9) You're building a simple calculator but not with code. You're using pure logic and flowcharting to decide what the machine should do based on the user's inputs.

Imagine this as an automated kiosk where a user:

1. Enters two numbers
2. Chooses an operation: +, -, *, or /
3. The system shows the result
4. If an invalid operation is chosen, show an error.

This is your chance to design the brain behind the machine using a clear, visual algorithm.

**Your Task**

Design a flowchart that performs:

- Addition
- Subtraction
- Multiplication

- Division

based on the user's choice.

**What Your Flowchart Must Include:**

- Input 1st number
- Input 2nd number
- Ask: Which operation?
- Branches for:
    - + → Add and show result
    - - → Subtract and show result
    - * → Multiply and show result
    - / → Divide (include a check for divide-by-zero)
- If invalid operation → Show error message
- End

**This is the pseudcode of the above question, try to make a flowchart in Flowgorithm.**

```
Start
  Output "Enter first number:"
  Input num1
  Output "Enter second number:"
  Input num2
  Output "Enter operation (+, -, *, /):"
  Input op
  If op == "+"
    Set result = num1 + num2
    Output "Result = ", result
  Else If op == "-"
    Set result = num1 - num2
```

```
    Output "Result = ", result
  Else If op == "*"
    Set result = num1 * num2
    Output "Result = ", result
  Else If op == "/"
    If num2 == 0
      Output "Error: Cannot divide by zero"
    Else
      Set result = num1 / num2
      Output "Result = ", result
    End If

  Else
    Output "Invalid Operation"
  End If
End.
```

---

[HARD] Q10) Imagine you're inside a magical mirror maze.

Each mirror shows another version of you, standing in front of another mirror… and that one sees another… and another…

Suddenly, you realize a pattern:
Every “you” is doing the same thing, just at a different level.

That’s what recursion is: a process that calls itself, but with a slightly smaller problem each time.

**What is Recursion?**

**Recursion** is a method where the solution to a problem depends on solutions to smaller instances of the **same problem**.

**How it works**

1. **Base Case**:
   The stopping point where the problem is simple enough to answer directly
2. **Recursive Case**:
   Break the bigger problem into a smaller version of itself
3. **Unfolding**:
   Once you hit the base, answers start building back up

**Classic Example: Factorial**

The factorial of 4 is:

$4! = 4 \times 3 \times 2 \times 1 = 24$

But instead of multiplying directly, think recursively:

$4! = 4 \times 3!$

$3! = 3 \times 2!$

$2! = 2 \times 1!$

$1! = 1$ ← base case

**Steps for Solving Any Recursive Problem**

1. Identify the base case (smallest, simplest input)
2. Figure out how to reduce the problem
3. Combine results from smaller calls
4. Make sure it eventually **stops** (no infinite recursion)

You're teaching a robot to count down from a number to 1 but the robot **cannot use loops** (no for, no while). Instead, it can call itself to finish the task.

**Your Tasks:**

1. Describe in your own words:
   What would the robot do at each step if you said "Count down from 5"?
2. Write **pseudocode** using recursion that prints numbers from n to 1
3. Make sure your logic **stops** at the right moment

4. Add a condition to avoid **negative numbers**

---

[MEDIUM] Q11) Design a Flowgorithm flowchart that calculates and displays the sine values for angles starting from 0° to 360°, increasing by 10° each step.

**Instructions:**

1. Use a **loop structure** (preferably For loop).
2. Convert each angle from **degrees to radians** using the formula:

$$\text{Radians} = (\text{angle} \times \pi)/180$$

3. Display both the **angle in degrees** and the **sine value**.
4. End the program after reaching 360°.