# Classroom LAB Assignment- 1

## Question 1: Sensor Data Analysis (1-D Array)

**Scenario**

A smart factory records the number of defective items produced every hour.
The system stores this data in an integer array. You are asked to process the data and print specific results.

**Task**

Write a C program that:

1. Stores the hourly defective counts in an array.
2. Calculates the **sum of defects at even indices**.
3. Prints the **difference between the maximum and minimum values** in the array.

**Given Data**

int defects[] = {12, 7, 9, 15, 6, 10};

**Questions**

1. What C code will you write to perform the above tasks?
2. What will be the **output printed** by the program?

**Expected Output**

Sum of defects at even indices: 27

Difference between max and min: 9

## Question 2: Student Marks Processing (Array Manipulation)

**Scenario**

A university stores marks of students in a one-dimensional array.
The system applies a moderation rule where bonus marks are added based on index position.

**Task**

Write a C program that:

1. Stores marks in an integer array.
2. Adds **5 marks** to elements stored at **odd indices**.
3. Prints the final array.

**Given Code Skeleton**

int marks[] = {60, 70, 80, 90, 50};

int i;

**Questions**

1. Complete the C code.
2. What will be the final output?

**Expected Output**

60 75 80 95 50

## Question 3: Network Packet Reordering (Array Index Logic)

**Scenario**

In a networking application, packet IDs are stored in an array.
Due to transmission delay, the packets need to be processed in **reverse order**, but only for a specific range.

**Task**

Write a C program that:

1. Stores packet IDs in an array.
2. Prints elements from index **1 to 4 in reverse order** (inclusive).

**Given Data**

int packets[] = {101, 202, 303, 404, 505, 606};

**Questions**

1. Write the required C code.
2. What will be printed?

**Expected Output**

505 404 303 202

---

## Question 4: Memory Optimization Check (Array & Conditional Logic)

**Scenario**

An operating system module tracks memory block usage.
A block is considered **overloaded** if its value is greater than the average usage.

**Task**

Write a C program that:

1. Stores memory usage values in an array.
2. Calculates the average.

3. Prints all values **greater than the average**.

**Given Data**

int memory[] = {20, 40, 60, 80};

**Expected Output**

60 80