

OOP Assignment # 1 (100 Marks)

General Instructions:

- Write clean, proper indentations, well-**commented** and easy to understand code.
- Use proper dynamic memory management—allocate and **free memory** as required.
- Ensure your code compiles without errors and runs correctly.
- Submit your handwritten assignment by the due date.
- Also submit your assignment on GCR(make pdf by taking pictures of your assignment)

Due Date: 20/02/2025

Question 1: Advanced Array Manipulation [20 Marks]

1. Array Initialization & Pointers:

- Declare an array of integers (size of 10). Initialize the array with values.
- Declare a pointer that points to the first element of the array.

2. Operations:

- **Reverse Array:** Write a function that reverses the array in place using pointer arithmetic.
- **Find Second Largest Element:** Write a function that finds and returns the second largest element in the array using pointers.
- **Rotate Array:** Implement a function that rotates the array to the right by k positions (where k is an integer input by the user).

3. Output:

- Display the original array.
 - Display the reversed array.
 - Display the second largest element.
 - Display the array after rotation.
-

Question 2: Dynamic Matrix Operations [20 Marks]

1. Matrix Allocation & Initialization:

- Prompt the user to enter the size n of an n x n matrix.
- Dynamically allocate memory for the matrix and fill it with random values between 1 and 100.

2. Matrix Operations:

- **Display Matrix:** Write a function to display the matrix in a neat n x n format.
- **Diagonal Sums:** Write a function to calculate and return the sum of the main diagonal elements.
- **Absolute Difference of Diagonals:** Write a function to calculate the sum of the secondary diagonal and compute the absolute difference between the two diagonal sums.

3. Output:

- Display the matrix.
- Display the sum of the main and secondary diagonals.

- Display the absolute difference between the two diagonal sums.

Question 3: Student Grade Tracker [20 Marks]

1. Define the Student Struct:

- Create a struct Student that contains:
 - A character array name[50] for the student's name.
 - An integer numSubjects representing the number of subjects.
 - A pointer to dynamically allocated array scores[] that stores the student's scores.

2. Operations:

- **Input Data:** Prompt the user for the number of students and their details (name, number of subjects, and scores).
- **Calculate Average:** Write a function to calculate the average score of each student.
- **Sort by Average:** Write a function to sort all students in descending order of their average scores.

3. Output:

- Display each student's name and average score.
- Display the student with the highest average score.

Question 4: Recursive Sum and Factorial [20 Marks]

1. Array Operations with Recursion:

- Declare an array of integers (size 5) and initialize it with values.
- Implement a recursive function sumArray to calculate the sum of all elements in the array.

2. Factorial Computation:

- Implement a recursive function factorial(n) to compute the factorial of an integer n.

3. Output:

- Display the sum of the array elements.
- Display the factorial of the length of the array (i.e., factorial of 5).

Question 5: Pointer and Structure Operations [20 Marks]

1. Define the Point Struct:

- Create a struct Point that contains:
 - Two integers x and y representing the coordinates of a point.

2. Operations:

- **Input Data:** Prompt the user for the number of points. Dynamically allocate memory to store the points.
- **Translate Points:** Write a function that accepts a pointer to a Point and translates each point by a given offset dx and dy (i.e., shift the points by dx and dy in the x and y directions).
- **Print Points:** Write a function that prints all the points.

3. Output:

- Display the list of points before and after translation.