DSA LAB 01

Name: *Syed Wahaaj Ali*
Roll No: *CT-24035*
Course Code: *CT-159*
CSIT
Section: *A*

Q1:

1. Write a C++ program to copy data of a 2D array in a 1D array using Column Major Order.

Code:

```cpp
/*
    Syed Wahaaj Ali
    CT-24035
*/
#include <iostream>
using namespace std;

int main() {
    int rows, cols;

    cout << "Enter number of rows: ";
    cin >> rows;
    cout << "Enter number of columns: ";
    cin >> cols;

    int** arr = new int*[rows];
    for (int i = 0; i < rows; i++) {
        arr[i] = new int[cols];
    }
    int* oneD = new int[rows * cols];

    cout << "Enter elements of 2D array:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cin >> arr[i][j];
        }
    }

    int index = 0;
    for (int j = 0; j < cols; j++) {
        for (int i = 0; i < rows; i++) {
            oneD[index++] = arr[i][j];
        }
    }

    cout << "1D array in column-major order:" << endl;
    for (int i = 0; i < rows * cols; i++) {
        cout << oneD[i] << " ";
    }
    cout << endl;
```

```
    delete[] oneD;
    for (int i = 0; i < rows; i++) {
        delete[] arr[i];
    }
    delete[] arr;

    return 0;
}
```

Output:

```
Enter number of rows: 3
Enter number of columns: 3
Enter elements of 2D array:
2
3
1
4
5
6
7
8
9
1D array in column-major order:
2 4 7 3 5 8 1 6 9


=== Code Execution Successful ===
```

Q2:

2. Write a program to calculate the GPA of students of all subjects of a single semester . Assume all the courses have the same credit hour (let's assume 3 credit hours).

	Data Structure	Programming for AI	Digital Logic Design	Probability & Statistics	Finance & Accounting
Ali	3.66	3.33	4.0	3.0	2.66
Hiba	3.33	3.0	3.66	3.0	---
Asma	4.0	3.66	2.66	---	---
Zain	2.66	2.33	4.0	---	---
Faisal	3.33	3.66	4.0	3.0	3.33

Code:

```
/*
    Syed Wahaaj Ali
    CT-24035
*/


#include <iostream>
#include <iomanip>
using namespace std;

float* gpa(float **arr, int sub, int students) {
    float* res = new float[students];
    for (int i = 0; i < students; i++) {
        res[i] = 0;
        for (int j = 0; j < sub; j++) {
            res[i] += arr[i][j];
        }
        res[i] /= sub;
    }
    return res;
}

int main() {
    cout << fixed << setprecision(2);
    int fixedStudents = 5, fixedCourses = 5;

    float **grid = new float*[fixedStudents];
    for (int i = 0; i < fixedStudents; i++) {
        grid[i] = new float[fixedCourses];
    }

    double init[5][5] = {
        {3.66, 3.33, 4.0, 3.0, 2.66},   // Ali
        {3.33, 3.0,  3.66, 3.0, 0},     // Hiba
        {4.0,  3.66, 2.66, 0, 0},       // Asma
        {2.66, 2.33, 4.0,  0, 0},       // Zain
        {3.33, 3.66, 4.0,  3.0, 3.33}   // Faisal
    };

    for (int i = 0; i < fixedStudents; i++) {
```

```
        for (int j = 0; j < fixedCourses; j++) {
            grid[i][j] = init[i][j];
        }
    }

    string names[5] = {"Ali", "Hiba", "Asma", "Zain", "Faisal"};

    cout << "--- Fixed Test (Manual Data) ---\n";
    float *res1 = gpa(grid, fixedCourses, fixedStudents);
    for (int i = 0; i < fixedStudents; i++) {
        cout << names[i] << " GPA: " << res1[i] << endl;
    }
    delete[] res1;

    for (int i = 0; i < fixedStudents; i++) {
        delete[] grid[i];
    }
    delete[] grid;

    int c, s;
    cout << "\nEnter number of courses: ";
    cin >> c;
    cout << "Enter number of students: ";
    cin >> s;

    float **matrix = new float*[s];
    for (int i = 0; i < s; i++) {
        matrix[i] = new float[c];
    }

    cout << "\n--- Enter Grades ---\n";
    for (int i = 0; i < s; i++) {
        cout << "\nStudent " << i + 1 << ":\n";
        for (int j = 0; j < c; j++) {
            cout << "Course " << j + 1 << ": ";
            cin >> matrix[i][j];
        }
    }

    float *res = gpa(matrix, c, s);
    cout << "\n--- Custom Test Results ---\n";
    for (int i = 0; i < s; i++) {
        cout << "Student " << i + 1 << " GPA: " << res[i] << endl;
    }
    delete[] res;

    for (int i = 0; i < s; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;

    return 0;
}
```

Output:

```
--- Fixed Test (Manual Data) ---
Ali GPA: 3.33
Hiba GPA: 2.60
Asma GPA: 2.06
Zain GPA: 1.80
Faisal GPA: 3.46

Enter number of courses: 4
Enter number of students: 2

--- Enter Grades ---

Student 1:
Course 1: 3.5
Course 2: 2.98
Course 3: 3.10
Course 4: 3.21

Student 2:
Course 1: 4.00
Course 2: 3.99
Course 3: 3.19
Course 4: 2.97

--- Custom Test Results ---
Student 1 GPA: 3.20
Student 2 GPA: 3.54


=== Code Execution Successful ===
```

Q3:

3. The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.
   For example, for arr = [2,3,4], the median is 3.
   For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.
   Implement the MedianFinder class:
   - MedianFinder() initializes the MedianFinder object.
   - void addNum(int num) adds the integer num from the data stream to the data structure.
   - double findMedian() returns the median of all elements so far. Answers within 10-5 of the actual answer will be accepted.

   Example 1:
   Input: ["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]
   [[], [1], [2], [], [3], []]
   Output: [null, null, null, 1.5, null, 2.0]
   Explanation
   MedianFinder medianFinder = new MedianFinder();
   medianFinder.addNum(1); // arr = [1]
   medianFinder.addNum(2); // arr = [1, 2]
   medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)
   medianFinder.addNum(3); // arr[1, 2, 3]
   medianFinder.findMedian(); // return 2.0
   Constraints: -105 <= num <= 105
   There will be at least one element in the data structure before calling findMedian.
   At most 5 * 104 calls will be made to addNum and findMedian.

Code:

```
/*
    Syed Wahaaj Ali
    CT-24035
*/

#include <iostream>
#include <algorithm>
using namespace std;

class MedianFinder {
    private:
        int* arr;
        int size;

    public:
        MedianFinder() {
            arr = nullptr;
            size = 0;
        }

        void addNum(int num) {
            int* newArr = new int[size + 1];
            for (int i = 0; i < size; i++) {
                newArr[i] = arr[i];
            }
            newArr[size] = num;
```

```
            delete[] arr;
            arr = newArr;
            size++;
        }

        double findMedian() {
            int* temp = new int[size];
            for (int i = 0; i < size; i++) {
                temp[i] = arr[i];
            }
            sort(temp, temp + size);
            double median;
            if (size % 2 == 1) {
                median = temp[size / 2];
            } else {
                median = (temp[size / 2 - 1] + temp[size / 2]) / 2.0;
            }

            delete[] temp;
            return median;
        }

        ~MedianFinder() {
            delete[] arr;
        }
};

int main() {
    MedianFinder medianFinder;
    medianFinder.addNum(1);
    medianFinder.addNum(2);
    cout << medianFinder.findMedian() << endl;
    medianFinder.addNum(3);
    cout << medianFinder.findMedian() << endl;
    return 0;
}
```

Output:

```
1.5
2


=== Code Execution Successful ===
```

Q4:

4. Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1. You must write an algorithm with O(log n) runtime complexity.

   Example 1: Input: nums = [-1,0,3,5,9,12], target = 9, Output: 4

   Explanation: 9 exists in nums and its index is 4

   Example 2: Input: nums = [-1,0,3,5,9,12], target = 2, Output: -1

   Explanation: 2 does not exist in nums so return -1

   Constraints:

   1 <= nums.length <= 104

   -104 < nums[i], target < 104

   All the integers in nums are unique.

Code:

```
/*
    Syed Wahaaj Ali
    CT-24035
*/


#include <iostream>
using namespace std;

int search(int arr[], int size, int target) {
    int l = 0, r = size - 1, mid;
    while (l <= r) {
        mid = (l + r) / 2;
        if (arr[mid] == target) {
            return mid;
        }
        else if (arr[mid] > target) {
            r = mid - 1;
        }
        else {
            l = mid + 1;
        }
    }
    return -1;
}

int main() {
    int size;
    cout << "Enter the size of array: ";
    cin >> size;

    int* arr = new int[size];

    cout << "Enter " << size << " sorted elements:\n";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }

    int target;
```

```
    cout << "Enter element to look for: ";
    cin >> target;

    int idx = search(arr, size, target);
    if (idx == -1) {
        cout << "Element does not exist!!\n";
    }
    else {
        cout << "Element is at index " << idx << endl;
    }

    delete[] arr;
    return 0;
}
```

Output:

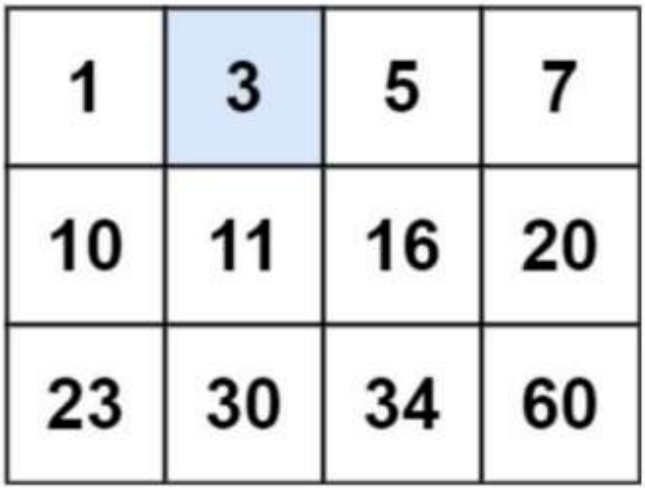
```
Enter the size of array: 5
Enter 5 sorted elements:
1
2
5
8
9
Enter element to look for: 5
Element is at index 2


=== Code Execution Successful ===
```

Q5:

5. nums is sorted in ascending order.You are given an m x n integer matrix with the following two properties: Each row is sorted in non-decreasing order. The first integer of each row is greater than the last integer of the previous row. Given an integer target, return true if target is in matrix or false otherwise. You must write a solution in O(log(m * n)) time complexity.

Example 1:



Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3, Output: true

Constraints:

m == matrix.length, n == matrix[i].length, 1 <= m, n <= 100, -104 <= matrix[i][j], target <= 104

Code:

```cpp
/*
    Syed Wahaaj Ali
    CT-24035
*/

#include <iostream>
#include <vector>
using namespace std;

bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int rows = matrix.size();
    int cols = matrix[0].size();

    int start = 0, end = rows * cols - 1;

    while (start <= end) {
        int mid = start + (end - start) / 2;
        int row = mid / cols;
        int col = mid % cols;

        if (matrix[row][col] == target) return true;
        else if (matrix[row][col] < target) start = mid + 1;
        else end = mid - 1;
    }
    return false;
}
```

```cpp
int main() {
    int rows, cols;
    cout << "Enter rows: ";
    cin >> rows;
    cout << "Enter cols: ";
    cin >> cols;

    vector<vector<int>> matrix(rows, vector<int>(cols));

    cout << "Enter matrix values (sorted row-wise):\n";
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cin >> matrix[i][j];
        }
    }

    int target;
    cout << "Enter target: ";
    cin >> target;

    if (searchMatrix(matrix, target))
        cout << "Found\n";
    else
        cout << "Not Found\n";

    return 0;
}
```

Output:

```
Enter rows: 4
Enter cols: 3
Enter matrix values (sorted row-wise):
1
3
5
7
10
11
16
20
23
30
34
60
Enter target: 3
Found


=== Code Execution Successful ===
```