Data Structure Algorithm & Application (CT-159)
Lab 01

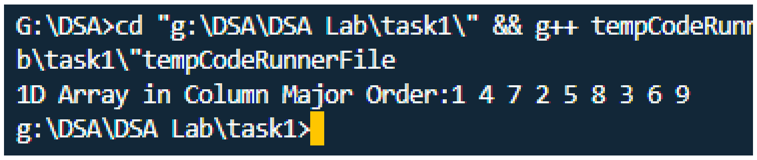
Exercise

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

Code:

```
#include <iostream>
using namespace std;
int main() {
    int rows=3;
    int cols=3;
    int index=0;
int array2D[rows][cols]={{1,2,3},{4,5,6},{7,8,9}};
int array1D[9];
    for(int col =0;col<cols;col++){
        for(int row =0 ;row<rows;row++){
        array1D[index]=array2D[row][col];
        index++;
    }
}
cout <<"1D Array in Column Major Order:";
for (int i = 0; i < rows * cols; i++) {
cout << array1D[i] <<" ";
}
return 0;
}
```

Output:



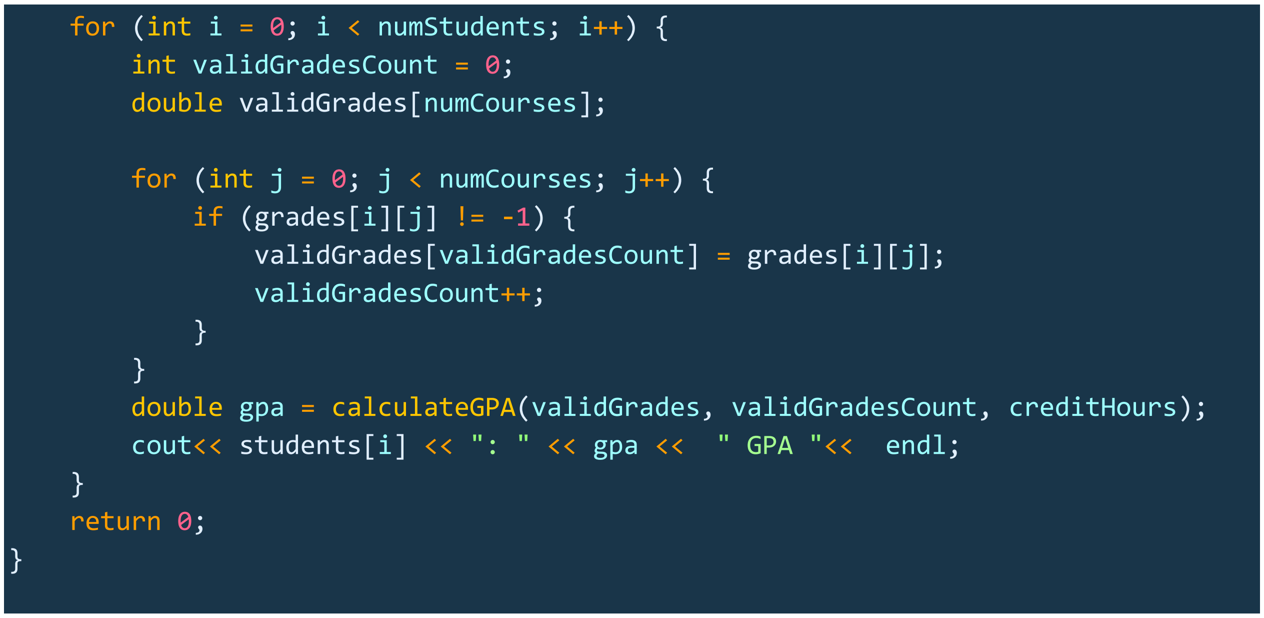
```
G:\DSA>cd "g:\DSA\DSA Lab\task1\" && g++ tempCodeRunn
b\task1\"tempCodeRunnerFile
1D Array in Column Major Order:1 4 7 2 5 8 3 6 9
g:\DSA\DSA Lab\task1>
```

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

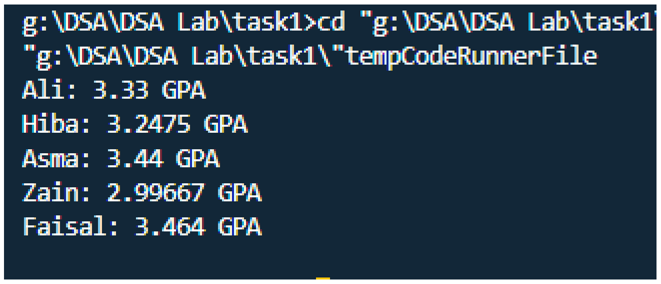
Code:

```
#include <iostream>
#include <string>
using namespace std;

double calculateGPA(double grades[], int numberOfGrades, int creditHours) {
    double totalPoints = 0;
    for (int i = 0; i < numberOfGrades; i++) {
        totalPoints += grades[i] * creditHours;
    }
    return totalPoints / (numberOfGrades * creditHours);
}
int main() {
    const int numStudents = 5;
    const int numCourses = 5;
    const int creditHours = 3;

    string students[numStudents] = {"Ali", "Hiba", "Asma", "Zain", "Faisal"};

    double grades[numStudents][numCourses] = {
        {3.66, 3.33, 4.0, 3.0, 2.66},
        {3.33, 3.0, 3.66, 3.0, -1},
        {4.0, 3.66, 2.66, -1, -1},
        {2.66, 2.33, 4.0, -1, -1},
        {3.33, 3.66, 4.0, 3.0, 3.33}
    };
```



```
    for (int i = 0; i < numStudents; i++) {
        int validGradesCount = 0;
        double validGrades[numCourses];

        for (int j = 0; j < numCourses; j++) {
            if (grades[i][j] != -1) {
                validGrades[validGradesCount] = grades[i][j];
                validGradesCount++;
            }
        }
        double gpa = calculateGPA(validGrades, validGradesCount, creditHours);
        cout<< students[i] << ": " << gpa <<  " GPA "<<  endl;
    }
    return 0;
}
```

Output:



```
g:\DSA\DSA Lab\task1>cd "g:\DSA\DSA Lab\task1
"g:\DSA\DSA Lab\task1\"tempCodeRunnerFile
Ali: 3.33 GPA
Hiba: 3.2475 GPA
Asma: 3.44 GPA
Zain: 2.99667 GPA
Faisal: 3.464 GPA
```

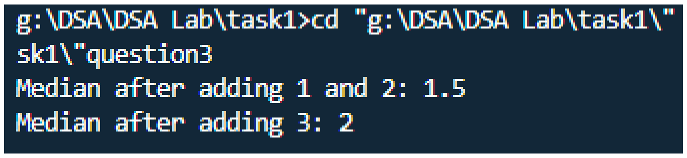
3. The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.
For example, for arr = [2,3,4], the median is 3.
For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.
Implement the MedianFinder 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: [&quot;MedianFinder&quot;, &quot;addNum&quot;, &quot;addNum&quot;, &quot;findMedian&quot;, &quot;addNum&quot;,
&quot;findMedian&quot;]
[[], [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 &lt;= num &lt;= 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:

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

class MedianFinder {
private:
    int* nums;
    int size;
    int capacity;
public:
    MedianFinder() {
        size = 0;
        capacity = 1;
        nums = new int[capacity];
    }
    ~MedianFinder() {
        delete[] nums;
    }
    void addNum(int num) {
        if (size == capacity) {
            int* temp = new int[capacity * 2];
            capacity *= 2;
            for (int i = 0; i < size; i++) {
                temp[i] = nums[i];
            }
            delete[] nums;
            nums = temp;
        }
        int i = size - 1;
        while (i >= 0 && nums[i] > num) {
            nums[i + 1] = nums[i];
            i--;
        }
        nums[i + 1] = num;
        size++;
    }
    double findMedian() {
        if (size % 2 == 0) {
            return (nums[size / 2 - 1] + nums[size / 2]) / 2.0;
        } else {
            return nums[size / 2];
        }
    }
};
int main() {
    MedianFinder medianFinder;
    medianFinder.addNum(1);
    medianFinder.addNum(2);
    cout << "Median after adding 1 and 2: " << medianFinder.findMedian() << endl;
    medianFinder.addNum(3);
    cout << "Median after adding 3: " << medianFinder.findMedian() << endl;
    return 0;
}
```

Output:



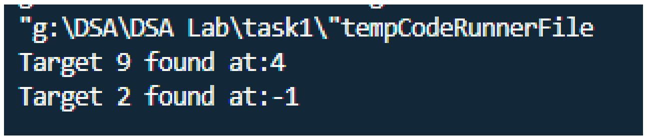
```
g:\DSA\DSA Lab\task1>cd "g:\DSA\DSA Lab\task1\"
sk1\"question3
Median after adding 1 and 2: 1.5
Median after adding 3: 2
```

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

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

int findTarget(int arr[], int target, int size) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
    int mid = (left + right) / 2;
    if (arr[mid] == target) {
        return mid;
    }
    if (arr[mid]<target){
        left = mid + 1;
    }
    else {
        right = mid - 1;
    }
    }
    return -1;
}

int main() {
    int arr1[] = {-1,0,3,5,9,12};
    int size1 = sizeof(arr1)/sizeof(arr1[0]);
    cout <<"Target "<<9<<" found at:"<< findTarget(arr1,9,size1) << endl;

    int arr2[] = {-1,0,3,5,9,12};
    int size2 = sizeof(arr2)/sizeof(arr2[0]);
    cout<<"Target "<<2<<" found at:"<< findTarget(arr2,2,size2) << endl;

    return 0;}
```

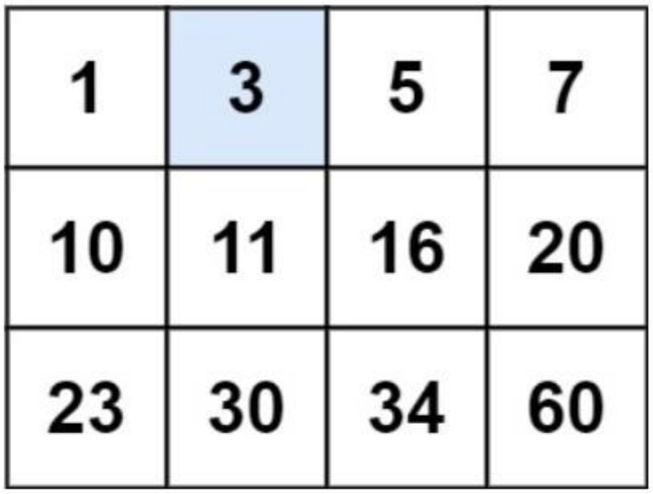
Output:



```
"g:\DSA\DSA Lab\task1\"tempCodeRunnerFile
Target 9 found at:4
Target 2 found at:-1
```

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:



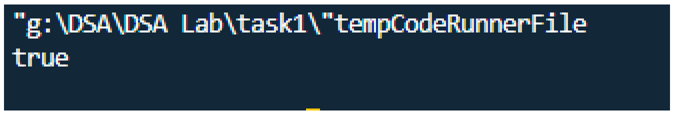
1	3	5	7
10	11	16	20
23	30	34	60

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:

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

bool searchMatrix(int matrix[][4], int m, int n, int target) {
    if (m == 0 || n == 0) return false;
    int left = 0;
    int right = m * n - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int mid_value = matrix[mid / n][mid % n];
        if (mid_value == target) {
            return true;
        } else if (mid_value < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return false;
}
int main() {
    int matrix[3][4] = {{1, 3, 5, 7},
                        {10, 11, 16, 20},
                        {23, 30, 34, 60}};

    int target = 3;
    bool result = searchMatrix(matrix, 3, 4, target);
    cout << (result ? "true" : "false") << endl;
    return 0;
}
```

Output:



```
"g:\DSA\DSA Lab\task1\"tempCodeRunnerFile
true
```

Lab 01 Evaluation		
Student Name: Adil Javed	**Student ID:**SE-23025	**Date: 2-Sep-2024**
Rubric	**Marks (25)**	**Remarks by teacher in accordance with the rubrics**
R1		
R2		
R3		
R4		
R5		