DSAA – Lab 01

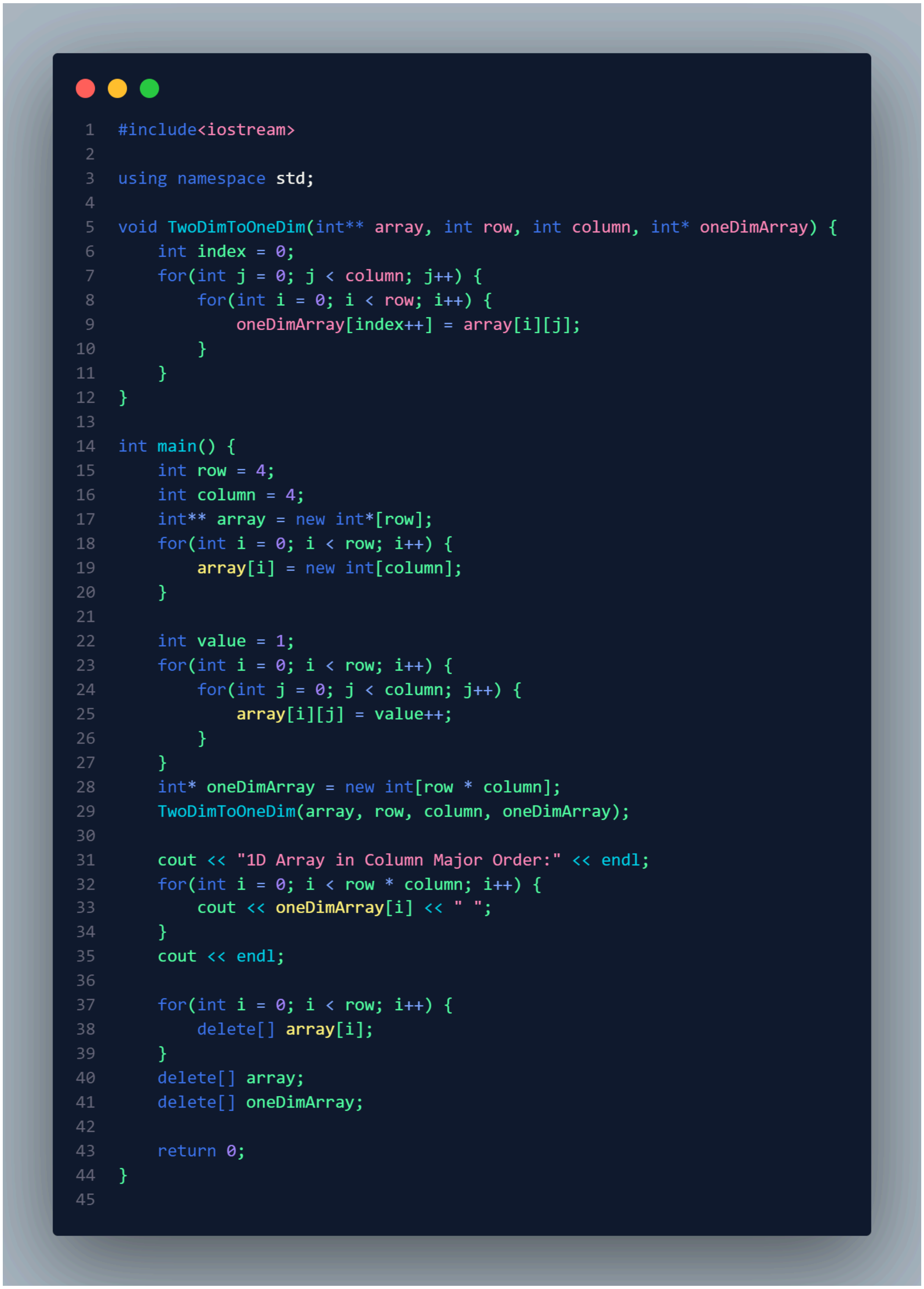
Fatimaa khan (ct-23024)
Rania imran (ct-23005)

BCIT | Section A

Question 1

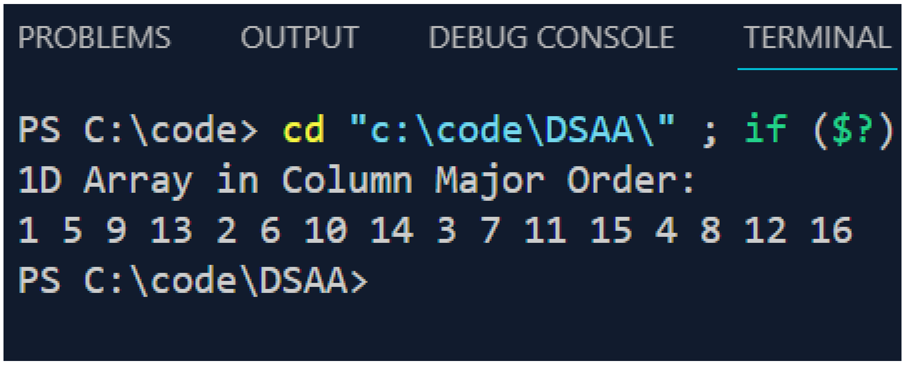
Write a C++ program to copy data of a 2D array in a 1D array using Column Major Order.

Source Code



```cpp
#include<iostream>

using namespace std;

void TwoDimToOneDim(int** array, int row, int column, int* oneDimArray) {
    int index = 0;
    for(int j = 0; j < column; j++) {
        for(int i = 0; i < row; i++) {
            oneDimArray[index++] = array[i][j];
        }
    }
}

int main() {
    int row = 4;
    int column = 4;
    int** array = new int*[row];
    for(int i = 0; i < row; i++) {
        array[i] = new int[column];
    }

    int value = 1;
    for(int i = 0; i < row; i++) {
        for(int j = 0; j < column; j++) {
            array[i][j] = value++;
        }
    }
    int* oneDimArray = new int[row * column];
    TwoDimToOneDim(array, row, column, oneDimArray);

    cout << "1D Array in Column Major Order:" << endl;
    for(int i = 0; i < row * column; i++) {
        cout << oneDimArray[i] << " ";
    }
    cout << endl;

    for(int i = 0; i < row; i++) {
        delete[] array[i];
    }
    delete[] array;
    delete[] oneDimArray;

    return 0;
}

```

Output

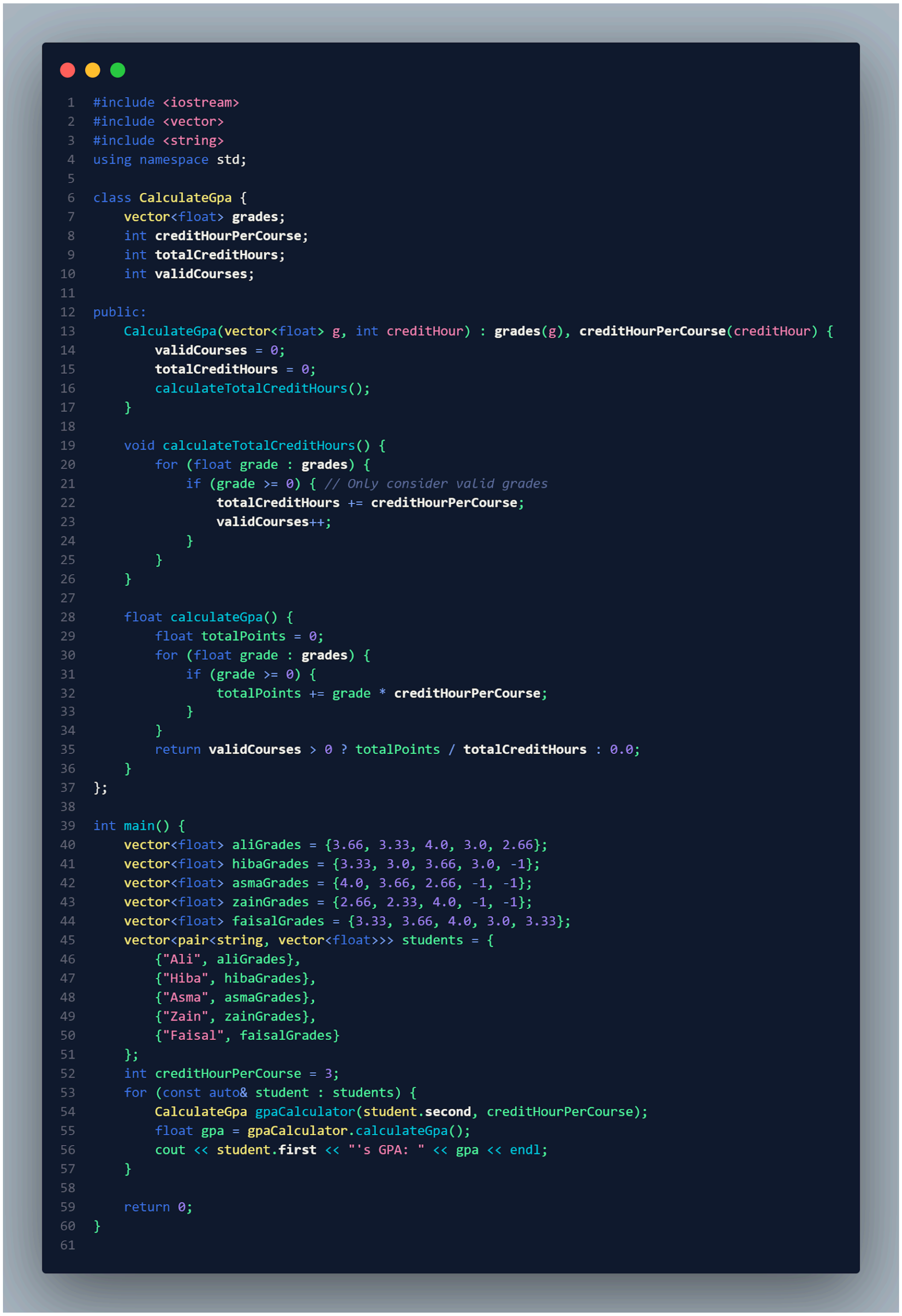


```
PS C:\code> cd "c:\code\DSAA\" ; if ($?)
1D Array in Column Major Order:
1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
PS C:\code\DSAA>
```

Question 2

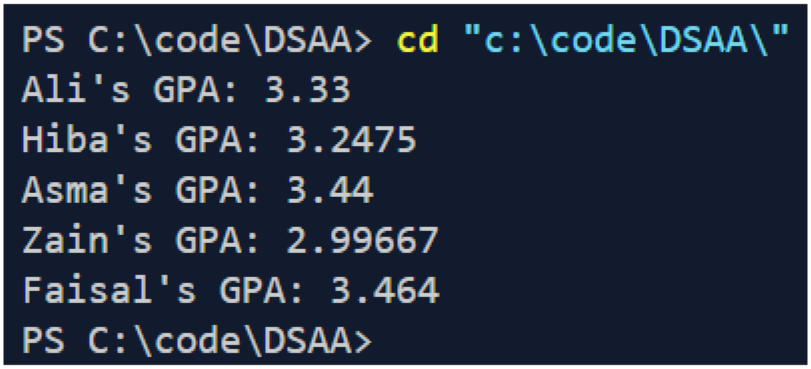
Write a program to calculate the GPA of students of all subjects of a single semester . Assume all the courses have the same credit hour (let's assume 3 credit hours).

Source Code



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

class CalculateGpa {
    vector<float> grades;
    int creditHourPerCourse;
    int totalCreditHours;
    int validCourses;

public:
    CalculateGpa(vector<float> g, int creditHour) : grades(g), creditHourPerCourse(creditHour) {
        validCourses = 0;
        totalCreditHours = 0;
        calculateTotalCreditHours();
    }

    void calculateTotalCreditHours() {
        for (float grade : grades) {
            if (grade >= 0) { // Only consider valid grades
                totalCreditHours += creditHourPerCourse;
                validCourses++;
            }
        }
    }

    float calculateGpa() {
        float totalPoints = 0;
        for (float grade : grades) {
            if (grade >= 0) {
                totalPoints += grade * creditHourPerCourse;
            }
        }
        return validCourses > 0 ? totalPoints / totalCreditHours : 0.0;
    }
};

int main() {
    vector<float> aliGrades = {3.66, 3.33, 4.0, 3.0, 2.66};
    vector<float> hibaGrades = {3.33, 3.0, 3.66, 3.0, -1};
    vector<float> asmaGrades = {4.0, 3.66, 2.66, -1, -1};
    vector<float> zainGrades = {2.66, 2.33, 4.0, -1, -1};
    vector<float> faisalGrades = {3.33, 3.66, 4.0, 3.0, 3.33};
    vector<pair<string, vector<float>>> students = {
        {"Ali", aliGrades},
        {"Hiba", hibaGrades},
        {"Asma", asmaGrades},
        {"Zain", zainGrades},
        {"Faisal", faisalGrades}
    };
    int creditHourPerCourse = 3;
    for (const auto& student : students) {
        CalculateGpa gpaCalculator(student.second, creditHourPerCourse);
        float gpa = gpaCalculator.calculateGpa();
        cout << student.first << "'s GPA: " << gpa << endl;
    }

    return 0;
}
```

Output



```
PS C:\code\DSAA> cd "c:\code\DSAA\"
Ali's GPA: 3.33
Hiba's GPA: 3.2475
Asma's GPA: 3.44
Zain's GPA: 2.99667
Faisal's GPA: 3.464
PS C:\code\DSAA>
```

Question 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]

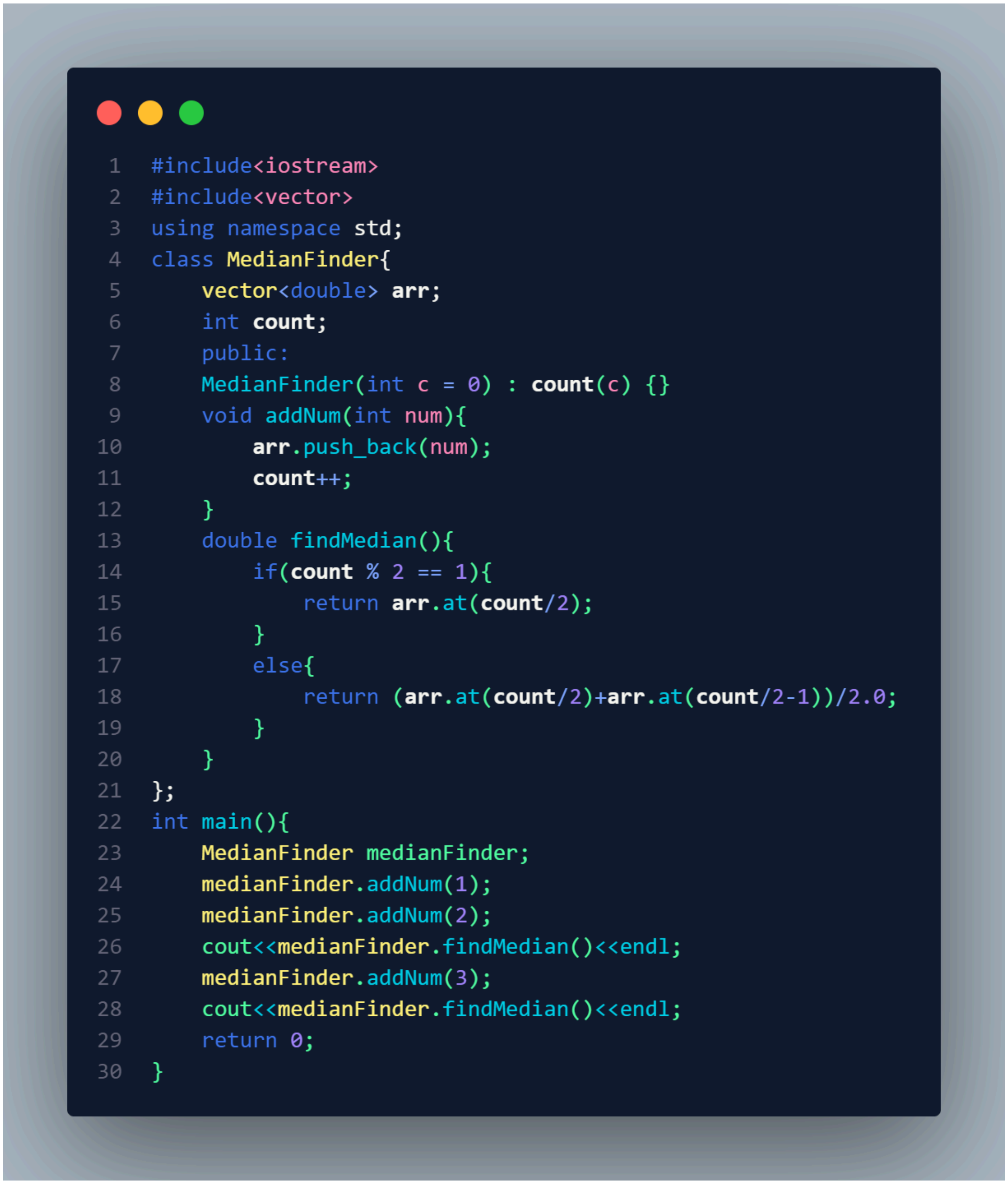
medianFinder.findMedian(); // return 2.0

Constraints: -105 <= num <= 105

There will be at least one element in the data structure before calling findMedian.

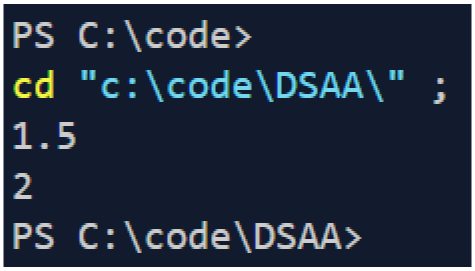
At most 5 * 104 calls will be made to addNum and findMedian.

Source Code



```cpp
#include<iostream>
#include<vector>
using namespace std;
class MedianFinder{
    vector<double> arr;
    int count;
    public:
    MedianFinder(int c = 0) : count(c) {}
    void addNum(int num){
        arr.push_back(num);
        count++;
    }
    double findMedian(){
        if(count % 2 == 1){
            return arr.at(count/2);
        }
        else{
            return (arr.at(count/2)+arr.at(count/2-1))/2.0;
        }
    }
};
int main(){
    MedianFinder medianFinder;
    medianFinder.addNum(1);
    medianFinder.addNum(2);
    cout<<medianFinder.findMedian()<<endl;
    medianFinder.addNum(3);
    cout<<medianFinder.findMedian()<<endl;
    return 0;
}
```

Ouput



```
PS C:\code>
cd "c:\code\DSAA\" ;
1.5
2
PS C:\code\DSAA>
```

Question 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

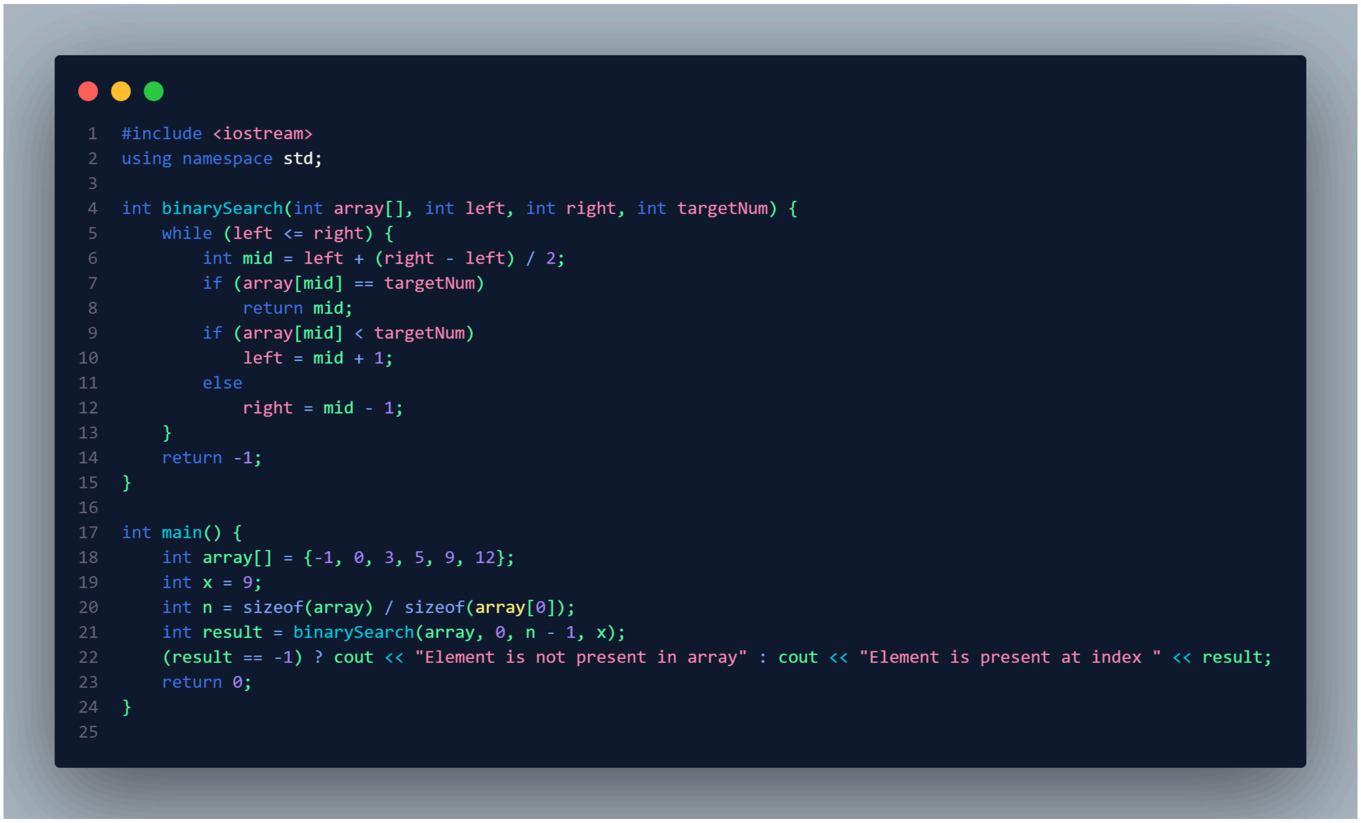
Constraints:

1 <= nums.length <= 104

-104 < nums[i], target < 104

All the integers in nums are unique.

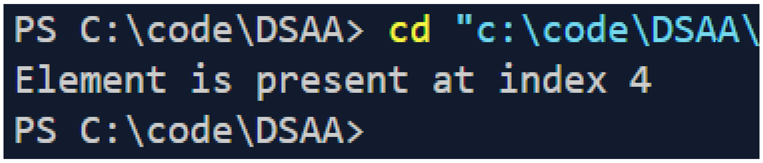
Source Code



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

int binarySearch(int array[], int left, int right, int targetNum) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (array[mid] == targetNum)
            return mid;
        if (array[mid] < targetNum)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}

int main() {
    int array[] = {-1, 0, 3, 5, 9, 12};
    int x = 9;
    int n = sizeof(array) / sizeof(array[0]);
    int result = binarySearch(array, 0, n - 1, x);
    (result == -1) ? cout << "Element is not present in array" : cout << "Element is present at index " << result;
    return 0;
}

```

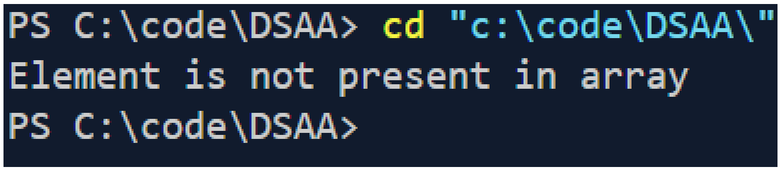
Output

When target==9



```
PS C:\code\DSAA> cd "c:\code\DSAA\
Element is present at index 4
PS C:\code\DSAA>
```

When target==2



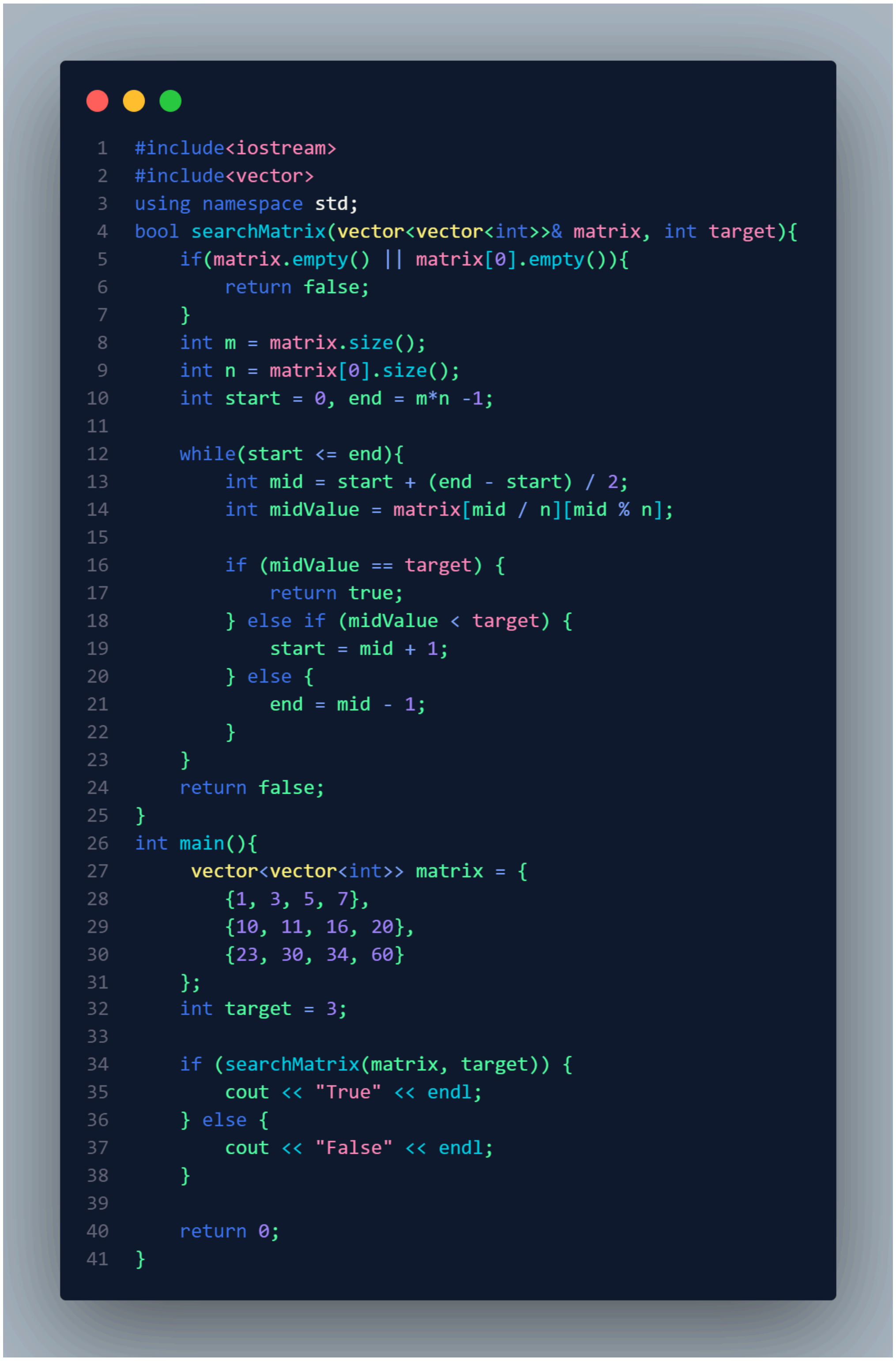
```
PS C:\code\DSAA> cd "c:\code\DSAA\"
Element is not present in array
PS C:\code\DSAA>
```

Question 5

You are given an m x n integer matrix with the

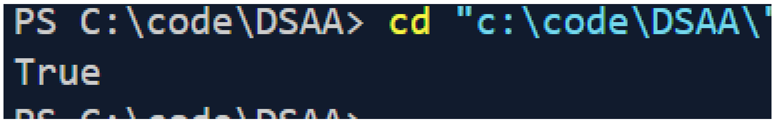
following two properties: Each row is sorted in non-decreasing order. The first integer of each row is greater than the last integer of the previous row. Given an integer target, return true if target is in matrix or false otherwise. You must write a solution in O(log(m * n)) time complexity.

Source Code



```cpp
#include<iostream>
#include<vector>
using namespace std;
bool searchMatrix(vector<vector<int>>& matrix, int target){
    if(matrix.empty() || matrix[0].empty()){
        return false;
    }
    int m = matrix.size();
    int n = matrix[0].size();
    int start = 0, end = m*n -1;

    while(start <= end){
        int mid = start + (end - start) / 2;
        int midValue = matrix[mid / n][mid % n];

        if (midValue == target) {
            return true;
        } else if (midValue < target) {
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return false;
}
int main(){
     vector<vector<int>> matrix = {
        {1, 3, 5, 7},
        {10, 11, 16, 20},
        {23, 30, 34, 60}
    };
    int target = 3;

    if (searchMatrix(matrix, target)) {
        cout << "True" << endl;
    } else {
        cout << "False" << endl;
    }

    return 0;
}
```

Output



```
PS C:\code\DSAA> cd "c:\code\DSAA\"
True
PS C:\code\DSAA>
```