**KASHAN BAIG**

**CT-24178**

**SEC D**

Q1.

void array2Dto1D(){

    int M=3,N=3;

    int \*\*arr = new int\*[N];

    for(int i=0;i<N;i++){

        arr[i] = new int[M];

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            arr[i][j] = i + j;

        }

    }

    cout << "2D Array:" << endl;

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            cout << arr[i][j] << " ";

        }

        cout << endl;

    }

    cout<< "Into 1D Array:"<<endl;

    for (int j = 0; j < M; j++) {

        for (int i = 0; i < N; i++) {

            cout << arr[i][j] << " ";

        }

    }

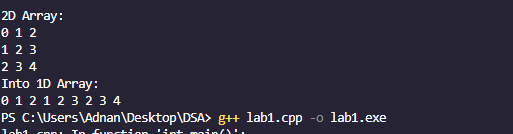
    for (int i = 0; i < N; i++) {

        delete[] arr[i];

    }

    delete[] arr;

}



Q2.

void calcGPA() {

    int no\_of\_students, no\_of\_subs;

    cout << "Enter the number of students: ";

    cin >> no\_of\_students;

    cout << "Enter the number of subjects: ";

    cin >> no\_of\_subs;

    string \*students = new string[no\_of\_students];

    string \*subjects = new string[no\_of\_subs];

    double \*studentGpa = new double[no\_of\_students];

    for (int i = 0; i < no\_of\_subs; i++) {

        cout << "Enter subject " << i + 1 << ": ";

        cin >> subjects[i];

    }

    for (int i = 0; i < no\_of\_students; i++) {

        cout << "Enter student " << i + 1 << ": ";

        cin >> students[i];

    }

    double \*\*arr2d = new double\*[no\_of\_students];

    for (int i = 0; i < no\_of\_students; i++) {

        arr2d[i] = new double[no\_of\_subs];

    }

    for (int row = 0; row < no\_of\_students; row++) {

        for (int col = 0; col < no\_of\_subs; col++) {

            cout << "Enter marks for Student: " << students[row]

                 << " -- Subject: " << subjects[col] << " --> ";

            cin >> arr2d[row][col];

        }

    }

    for (int i = 0; i < no\_of\_students; i++) {

        double total = 0;

        for (int j = 0; j < no\_of\_subs; j++) {

            total += arr2d[i][j];

        }

        studentGpa[i] = total / no\_of\_subs;

    }

    cout << "\nSubjects: ";

    for (int j = 0; j < no\_of\_subs; j++) {

        cout << subjects[j] << " ";

    }

    cout << endl;

    for (int i = 0; i < no\_of\_students; i++) {

        cout << "Student Name: " << students[i]

             << " has GPA of " << studentGpa[i] << endl;

    }

    cout << "\nDeleting memory..." << endl;

    for (int row = 0; row < no\_of\_students; row++) {

        delete[] arr2d[row];

    }

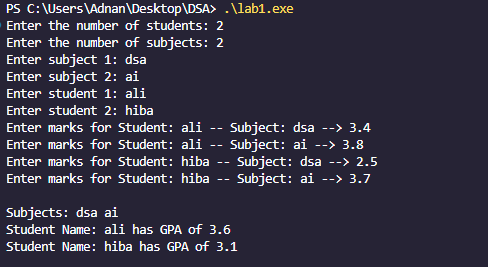
    delete[] arr2d;

    delete[] students;

    delete[] subjects;

    delete[] studentGpa;

}



Q3.

class MedianFinder {

private:

    vector<int> nums;

public:

    MedianFinder() {

    }

    void addNum(int *num*) {

*// Find the position to insert using lower\_bound*

        auto it = lower\_bound(nums.begin(), nums.end(), *num*);

        nums.insert(it, *num*);

    }

    double findMedian() {

        int n = nums.size();

        if (n % 2 == 1) {

*//if odd*

            return nums[n/2];

        } else {

*//if even*

            return (nums[n/2 - 1] + nums[n/2]) / 2.0;

        }

    }

};

int main(){

    MedianFinder mdf ;

    mdf.addNum(4);

    mdf.addNum(1);

    mdf.addNum(2);

    mdf.addNum(3);

    cout <<  mdf.findMedian();



Q4.

int binarySearch(int \**arr*, int *size*, int *target*)

{

    int low = 0;

    int high = *size* - 1;

    int mid;

    while (high >= low)

    {

        mid = low + (high - low) / 2; *// safer than (low+high)/2*

        if (*arr*[mid] == *target*)

        {

            return mid; *// found*

        }

        else if (*target* > *arr*[mid])

        {

            low = mid + 1; *// search right*

        }

        else

        {

            high = mid - 1; *// search left*

        }

    }

    return -1; *// not found*

}

int main()

{

    int arr[] = {-1, 0, 3, 5, 9, 12};

    int size = sizeof(arr) / sizeof(arr[0]);

    int target = 2;

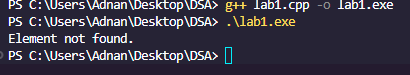
    int result = binarySearch(arr, size, target);

    if (result != -1)

        cout << "Element found at index: " << result << endl;

    else

        cout << "Element not found." << endl;



Q5.

bool binarySearchMatrix( int *arr*[][4], int *rows*, int *cols*, int *target*)

{

    int low = 0;

    int high = (*rows*\**cols*) - 1;

    int mid;

    while (high >= low)

    {

        mid = low + (high - low) / 2;

        int r = mid / *cols*;

        int c = mid % *cols*;

        if (*arr*[r][c] == *target*)

        {

            return true;

        }

        else if (*target* > *arr*[r][c])

        {

            low = mid + 1;

        }

        else

        {

            high = mid - 1;

        }

    }

    return false;

}

int main()

{

    int matrix[3][4] = {

        {1, 3, 5, 7},

        {10, 11, 16, 20},

        {23, 30, 34, 60}

    };

    int target = 16;

    if (binarySearchMatrix(matrix, 3, 4, target))

        cout << "Found";

    else

        cout << "Not Found";

