BINARY SEARCH CODING CHALLENGES

# 1.

Given a sorted array **arr[]** with possibly some duplicates, the task is to find the first and last occurrences of an element **x** in the given array.

**Note:** If the number **x** is not found in the array then return both the indices as -1.

**Examples:**

***Input :*** *arr[] = [1, 3, 5, 5, 5, 5, 67, 123, 125], x = 5*

***Output*** *: 2 5*

***Explanation:*** *First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5*

***Input :*** *arr[] = [1, 3, 5, 5, 5, 5, 7, 123, 125 ], x = 7*

***Output :*** *6 6*

***Explanation:*** *First and last occurrence of 7 is at index 6*

***Input:*** *arr[] = [1, 2, 3], x = 4*

***Output:*** *-1 -1*

# **Explanation:** No occurrence of 4 in the array, so, output is [-1, -1]

#include <iostream>

#include <vector>

using namespace std;

// Function to find the first occurrence of x

int findFirst(const vector<int>& arr, int x) {

    int low = 0, high = arr.size() - 1, result = -1;

    while (low <= high) {

        int mid = (low + high) / 2;

        if (arr[mid] == x) {

            result = mid;

            high = mid - 1; // Keep searching left

        } else if (arr[mid] < x) {

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    return result;

}

// Function to find the last occurrence of x

int findLast(const vector<int>& arr, int x) {

    int low = 0, high = arr.size() - 1, result = -1;

    while (low <= high) {

        int mid = (low + high) / 2;

        if (arr[mid] == x) {

            result = mid;

            low = mid + 1; // Keep searching right

        } else if (arr[mid] < x) {

            low = mid + 1;

        } else {

            high = mid - 1;

        }

    }

    return result;

}

int main() {

    vector<int> arr = {1, 3, 5, 5, 5, 5, 67, 123, 125};

    int x = 5;

    int first = findFirst(arr, x);

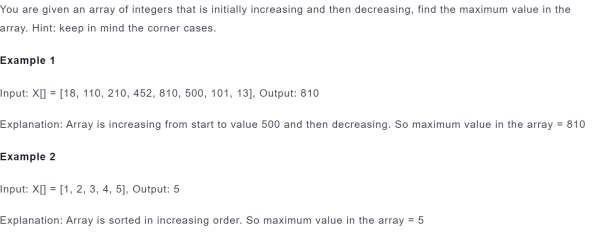
    int last = findLast(arr, x);

    cout << first << " " << last << endl;

    return 0;

}

**2.**

****

#include <iostream>

#include <vector>

using namespace std;

// Function to find maximum in a bitonic array

int findMaximum(const vector<int>& arr) {

    int low = 0, high = arr.size() - 1;

    while (low <= high) {

        int mid = (low + high) / 2;

        if ((mid == 0 || arr[mid] > arr[mid - 1]) &&

            (mid == arr.size() - 1 || arr[mid] > arr[mid + 1])) {

            return arr[mid];  // Peak element

        }

        else if (mid < arr.size() - 1 && arr[mid] < arr[mid + 1]) {

            low = mid + 1;  // Move to right

        }

        else {

            high = mid - 1; // Move to left

        }

    }

    return -1;

}

int main() {

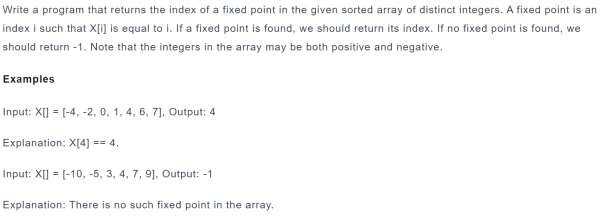
    vector<int> arr = {18, 110, 210, 452, 810, 500, 101, 13};

    int maxVal = findMaximum(arr);

    cout << "Maximum value: " << maxVal << endl;

    return 0;

}

****

**3.**

#include <iostream>

#include <vector>

using namespace std;

int findFixedPoint(const vector<int>& arr) {

    int low = 0, high = arr.size() - 1;

    while (low <= high) {

        int mid = (low + high) / 2;

        if (arr[mid] == mid) {

            return mid; // Fixed point found

        }

        else if (arr[mid] < mid) {

            low = mid + 1; // Search right side

        }

        else {

            high = mid - 1; // Search left side

        }

    }

    return -1; // No fixed point found

}

int main() {

    vector<int> arr1 = {-4, -2, 0, 1, 4, 6, 7};

    vector<int> arr2 = {-10, -5, 3, 4, 7, 9};

    cout << "Fixed point in arr1: " << findFixedPoint(arr1) << endl; // Output: 4

    cout << "Fixed point in arr2: " << findFixedPoint(arr2) << endl; // Output: -1

    return 0;

}