Lab task 7

Q1.

#include <iostream>

#include <vector>

Using namespace std;

// Function to perform binary search

Int binarySearch(const vector<int>& arr, int target) {

Int left = 0;

Int right = arr.size() – 1;

While (left <= right) {

// Calculate the middle index

Int mid = left + (right – left) / 2;

// Display the current active items

Cout << “Active items: “;

For (int I = left; I <= right; i++) {

Cout << arr[i] << (I < right ? “, “ : “”);

}

Cout << endl;

// Check if the target is at mid

If (arr[mid] == target) {

Return mid; // Target found

}

// If target is smaller than mid, it can only be in the left half

Else if (arr[mid] > target) {

Right = mid – 1;

}

// If target is larger than mid, it can only be in the right half

Else {

Left = mid + 1;

}

}

// Target not found

Return -1;

}

Int main() {

// Sample sorted array

Vector<int> arr = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

Int target;

// Input target value

Cout << “Enter the target value to search: “;

Cin >> target;

// Perform binary search

Int index = binarySearch(arr, target);

// Output the result

If (index != -1) {

Cout << “Target value “ << target << “ found at index: “ << index << endl;

} else {

Cout << “Target value “ << target << “ not found in the array.” << endl;

}

Return 0;

}

Q2.

#include <iostream>

#include <vector>

Using namespace std;

// Function to perform binary search to find the first occurrence of a target

Int findFirstOccurrence(const vector<int>& arr, int target) {

Int left = 0;

Int right = arr.size() – 1;

Int result = -1; // Variable to store the index of the first occurrence

While (left <= right) {

Int mid = left + (right – left) / 2;

// If the target is found, store the index and search in the left half

If (arr[mid] == target) {

Result = mid; // Update result with the current mid index

Right = mid – 1; // Move left to find the first occurrence

}

// If target is smaller, search in the left half

Else if (arr[mid] > target) {

Right = mid – 1;

}

// If target is larger, search in the right half

Else {

Left = mid + 1;

}

}

Return result; // Return the index of the first occurrence or -1 if not found

}

Int main() {

// Sample sorted array

Vector<int> arr = {1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9};

Int target;

// Input target value

Cout << “Enter the target value to search for its first occurrence: “;

Cin >> target;

// Find the first occurrence of the target

Int index = findFirstOccurrence(arr, target);

// Output the result

If (index != -1) {

Cout << “First occurrence of target value “ << target << “ found at index: “ << index << endl;

} else {

Cout << “Target value “ << target << “ not found in the array.” << endl;

}

Return 0;

}

Q3.

#include <iostream>

#include <vector>

Using namespace std;

// Function to perform binary search to find the last occurrence of a target

Int findLastOccurrence(const vector<int>& arr, int target) {

Int left = 0;

Int right = arr.size() – 1;

Int result = -1; // Variable to store the index of the last occurrence

While (left <= right) {

Int mid = left + (right – left) / 2;

// If the target is found, store the index and search in the right half

If (arr[mid] == target) {

Result = mid; // Update result with the current mid index

Left = mid + 1; // Move right to find the last occurrence

}

// If target is smaller, search in the left half

Else if (arr[mid] > target) {

Right = mid – 1;

}

// If target is larger, search in the right half

Else {

Left = mid + 1;

}

}

Return result; // Return the index of the last occurrence or -1 if not found

}

Int main() {

// Sample sorted array

Vector<int> arr = {1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9};

Int target;

// Input target value

Cout << “Enter the target value to search for its last occurrence: “;

Cin >> target;

// Find the last occurrence of the target

Int index = findLastOccurrence(arr, target);

// Output the result

If (index != -1) {

Cout << “Last occurrence of target value “ << target << “ found at index: “ << index << endl;

} else {

Cout << “Target value “ << target << “ not found in the array.” << endl;

}

Return 0;

}

Q4

#include <iostream>

#include <vector>

Using namespace std;

// Function to find the first occurrence of the target

Int findFirstOccurrence(const vector<int>& arr, int target) {

Int left = 0;

Int right = arr.size() – 1;

Int result = -1;

While (left <= right) {

Int mid = left + (right – left) / 2;

If (arr[mid] == target) {

Result = mid; // Update result with the current mid index

Right = mid – 1; // Move left to find the first occurrence

} else if (arr[mid] > target) {

Right = mid – 1;

} else {

Left = mid + 1;

}

}

Return result; // Return the index of the first occurrence or -1 if not found

}

// Function to find the last occurrence of the target

Int findLastOccurrence(const vector<int>& arr, int target) {

Int left = 0;

Int right = arr.size() – 1;

Int result = -1;

While (left <= right) {

Int mid = left + (right – left) / 2;

If (arr[mid] == target) {

Result = mid; // Update result with the current mid index

Left = mid + 1; // Move right to find the last occurrence

} else if (arr[mid] > target) {

Right = mid – 1;

} else {

Left = mid + 1;

}

}

Return result; // Return the index of the last occurrence or -1 if not found

}

Int main() {

// Sample sorted array

Vector<int> arr = {1, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9};

Int target;

// Input target value

Cout << “Enter the target value to search for its occurrences: “;

Cin >> target;

// Find the first and last occurrences of the target

Int firstIndex = findFirstOccurrence(arr, target);

Int lastIndex = findLastOccurrence(arr, target);

// Calculate the number of occurrences

If (firstIndex != -1 && lastIndex != -1) {

Int count = lastIndex – firstIndex + 1; // Number of occurrences

Cout << “The target value “ << target << “ occurs “ << count << “ times in the array.” << endl;

} else {

Cout << “Target value “ << target << “ not found in the array.” << endl;

}

Return 0;

}