LAB TASK 7

Name: Abdul Moiz

Sap I'd: 54482

Lab: DSA

From: CS-3-2

Task 1:

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

// Function to perform binary search and display active items at each stage
int binarySearch(int arr[], int size, int target) {
    int low = 0;
    int high = size - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        // Display the active items (current sub-array)
        cout << "Active items: ";
```

```
        for (int i = low; i <= high; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;

        if (arr[mid] == target) {
            return mid; // Target found
        }
        else if (arr[mid] < target) {
            low = mid + 1; // Search the right half
        }
        else {
            high = mid - 1; // Search the left half
        }
    }

    return -1; // Target not found
}

int main() {
    int arr[] = {2, 4, 6, 8, 10, 12, 14, 16};
    int size = sizeof(arr) / sizeof(arr[0]);
```

```
    int target = 10;

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

    if (result != -1) {
        cout << "Target found at index: " << result << endl;
    } else {
        cout << "Target not found" << endl;
    }

    return 0;
}
```

Task 2:

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

// Function to find the first occurrence of the target value using binary search
int findFirstOccurrence(int arr[], int size, int target) {
```

```
    int low = 0;
    int high = size - 1;
    int result = -1;  // To store the index of the first occurrence

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {
            result = mid;       // Update result and continue searching in the left half
            high = mid - 1;     // Move to the left half to find the first occurrence
        }
        else if (arr[mid] < target) {
            low = mid + 1;      // Search in the right half
        }
        else {
            high = mid - 1;     // Search in the left half
        }
    }

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

```
}

int main() {
    int arr[] = {2, 4, 4, 4, 8, 10, 12, 16};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 4;

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

    if (result != -1) {
        cout << "First occurrence of target found at index: " << result << endl;
    } else {
        cout << "Target not found" << endl;
    }

    return 0;
}
```

Task 3:

```
#include <iostream>
```

```
using namespace std;


// Function to find the last occurrence of the target value using binary search
int findLastOccurrence(int arr[], int size, int target) {
    int low = 0;
    int high = size - 1;
    int result = -1;  // To store the index of the last occurrence

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {
            result = mid;       // Update result and continue searching in the right half
            low = mid + 1;      // Move to the right half to find the last occurrence
        }
        else if (arr[mid] < target) {
            low = mid + 1;      // Search in the right half
        }
        else {
            high = mid - 1;     // Search in the left half
```

```
        }
    }

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

int main() {
    int arr[] = {2, 4, 4, 4, 8, 10, 12, 16};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 4;

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

    if (result != -1) {
        cout << "Last occurrence of target found at index: " << result << endl;
    } else {
        cout << "Target not found" << endl;
    }

    return 0;
}
```

Task 4:

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

// Function to find the first occurrence of the target value
int findFirstOccurrence(int arr[], int size, int target) {
    int low = 0;
    int high = size - 1;
    int result = -1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {
            result = mid;
            high = mid - 1;  // Continue searching in the left half
        }
        else if (arr[mid] < target) {
            low = mid + 1;
        }
        else {
```

```
            high = mid - 1;
        }
    }

    return result;
}

// Function to find the last occurrence of the target value
int findLastOccurrence(int arr[], int size, int target) {
    int low = 0;
    int high = size - 1;
    int result = -1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {
            result = mid;
            low = mid + 1;  // Continue searching in the right half
        }
        else if (arr[mid] < target) {
            low = mid + 1;
```

```
        }
        else {
            high = mid - 1;
        }
    }

    return result;
}

// Function to count the occurrences of the target value
int countOccurrences(int arr[], int size, int target) {
    int first = findFirstOccurrence(arr, size, target);
    if (first == -1) {
        return 0;  // Target not found
    }

    int last = findLastOccurrence(arr, size, target);
    return last - first + 1;
}

int main() {
    int arr[] = {2, 4, 4, 4, 8, 10, 12, 16};
```

```
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 4;

    int occurrences = countOccurrences(arr, size, target);

    if (occurrences > 0) {
        cout << "The target value " << target << " occurs " << occurrences
<< " times in the array." << endl;
    } else {
        cout << "The target value " << target << " is not found in the array."
<< endl;
    }

    return 0;
}
```