Binary_search.cpp

```cpp
/**
*    author:  devesh95
*
**/
#include <bits/stdc++.h>
using namespace std;

#define int             long long int
#define double          long double
#define F               first
#define S               second
#define pb              push_back
#define lb              lower_bound
#define ub              upper_bound
#define si              set <int>
#define vi              vector <int>
#define vvi             vector <vi>
#define pii             pair <int, int>
#define vpi             vector <pii>
#define mii             map <int, int>
#define sz(v)           ((int) v.size())
#define form(i, a, b)   for (int i=a; i<(b); i++)
#define forn(i, a)      for (int i=0; i<(a); i++)

////////////////////////////////////////////////////////
// Binary Search - Quick Notes
//
// Purpose: Efficiently search for an element or solve
//          problems requiring position or proximity in sorted arrays.
//
// ----------------------------------------------------
// Concepts:
// - Binary Search halves the search space in each iteration.
// - It works only on sorted arrays.
// - Time Complexity: O(log n)
//
// Applications:
// - Find a specific element in an array.
// - Find the element closest to a given value:
//     1. Closest to the left (max ≤ x).
//     2. Closest to the right (min ≥ x).
// - Count elements within a range efficiently.
//
// Tip: Binary search variants use different conditions for
//      updating the search range. Understanding these is key.
////////////////////////////////////////////////////////

// Standard Binary Search
int binarySearch(vi& arr, int x) {
    int low = 0, high = sz(arr) - 1;
    while (low <= high) {
```

```cpp
        int mid = (low + high) / 2;
        if (arr[mid] == x) return mid; // Element found
        else if (arr[mid] < x) low = mid + 1; // Search right
        else high = mid - 1; // Search left
    }
    return -1; // Element not found
}

// Closest to the Left (max ≤ x)
int closestLeft(vi& arr, int x) {
    int low = -1, high = sz(arr);
    while (low + 1 < high) {
        int mid = (low + high) / 2;
        if (arr[mid] <= x) low = mid; // Move left bound
        else high = mid; // Move right bound
    }
    return low; // Index of closest element ≤ x
}

// Closest to the Right (min ≥ x)
int closestRight(vi& arr, int x) {
    int low = -1, high = sz(arr);
    while (low + 1 < high) {
        int mid = (low + high) / 2;
        if (arr[mid] >= x) high = mid; // Move right bound
        else low = mid; // Move left bound
    }
    return high; // Index of closest element ≥ x
}

// Count numbers in range [l, r]
int countInRange(vi& arr, int l, int r) {
    int leftIdx = lb(arr.begin(), arr.end(), l) - arr.begin(); // First element ≥ l
    int rightIdx = ub(arr.begin(), arr.end(), r) - arr.begin(); // First element > r
    return rightIdx - leftIdx; // Count of elements in range [l, r]
}

void solve() {
    int n, x;
    cin >> n;
    vi arr(n);
    forn(i, n) cin >> arr[i];

    // Sort the array for binary search applications
    sort(arr.begin(), arr.end());

    int k;
    cin >> k;
    while (k--) {
        int l, r;
        cin >> l >> r;
        cout << countInRange(arr, l, r) << " ";
    }
    cout << endl;
```

```cpp

    // Example usage of other functions
    cin >> x;
    cout << "Binary Search Result: " << binarySearch(arr, x) << endl;
    cout << "Closest Left Index: " << closestLeft(arr, x) << endl;
    cout << "Closest Right Index: " << closestRight(arr, x) << endl;
}

int32_t main()
{
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);

#ifndef ONLINE_JUDGE
    freopen("input.txt",  "r",  stdin);
    freopen("output.txt", "w", stdout);
#endif
    clock_t z = clock();
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    cerr << "Run Time : " << ((double)(clock() - z) / CLOCKS_PER_SEC);
    return 0;
}

```