Name - Nikhil Purwar || Reg. No.- 12311508 || Course Code - 539

Solution 1 - First and Last Occurance

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

int firstOccurRec(const vector<int>& nums, int t, int first, int last)
{
    if (first > last) return -1;
    int mid = (first + last) / 2;
    if (nums[mid] == t)
    {
        int res = firstOccurRec(nums, t, first, mid - 1);
        return (res == -1) ? mid : res;
    }
    else if (nums[mid] < t)
    {
        return firstOccurRec(nums, t, mid + 1, last);
    }
    else
    {
        return firstOccurRec(nums, t, first, mid - 1);
    }
}
```

```cpp
int lastOccurRec(const vector<int>& nums, int t, int first, int last)
{
    if (first > last) return -1;
    int mid = (first + last) / 2;
    if (nums[mid] == t)
    {
        int res = lastOccurRec(nums, t, mid + 1, last);
        return (res == -1) ? mid : res;
    }
    else if (nums[mid] < t)
    {
        return lastOccurRec(nums, t, mid + 1, last);
    }
    else
    {
        return lastOccurRec(nums, t, first, mid - 1);
    }
}
```

```cpp
int main()
{
    vector<int> nums = {5, 7, 7, 8, 8, 10};
    int t = 8;
    cout << "First Occurrence is " << firstOccurRec(nums, t, 0, nums.size() - 1) << "\nLast Occurrence is "
<< lastOccurRec(nums, t, 0, nums.size() - 1) << endl;
    return 0;
}
```

Output -

```
First Occurrence is 3
Last Occurrence is 4
PS C:\Users\npnik\Desktop\CPP\CA>
```

Solution 2 - Count Inversions (Merge Sort)

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

int mergeAndCount(vector<int>& arr, int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;
    vector<int> leftArr(n1);
    vector<int> rightArr(n2);
    for (int i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
    rightArr[j] = arr[mid + 1 + j];
    int i = 0, j = 0, k = left, invCount = 0;
    while (i < n1 && j < n2)
    {
        if (leftArr[i] <= rightArr[j])
        {
            arr[k++] = leftArr[i++];
        }
        else
        {
            arr[k++] = rightArr[j++];
            invCount += (n1 - i);
        }
    }
    while (i < n1)
        arr[k++] = leftArr[i++];
    while (j < n2)
        arr[k++] = rightArr[j++];
    return invCount;
}
```

```cpp
int mergeSort(vector<int>& arr, int left, int right)
{
    int invCount = 0;
    if (left < right)
    {
        int mid = left + (right - left) / 2;
        invCount += mergeSort(arr, left, mid);
        invCount += mergeSort(arr, mid + 1, right);
        invCount += mergeAndCount(arr, left, mid, right);
    }
    return invCount;
}
```

```cpp
int main()
{
    vector<int> arr = {2, 3, 8, 6, 1};
    cout << "Number of inversions: " << mergeSort(arr, 0, arr.size() - 1) << endl;
    return 0;
}
```

Output-

```
Number of inversions: 5
PS C:\Users\npnik\Desktop\CPP\CA>
```