Count Inversions in C++

```
#include <iostream>
#include <vector>

using namespace std;

long long ans;

void merge(vector<long long>& arr, int l, int m, int
r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    vector<long long> L(n1), R(n2);

    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    int i = 0, j = 0, k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
            ans += (m - l + 1 - i);
        }
    }

    while (i < n1) {
        arr[k++] = L[i++];
    }

    while (j < n2) {
        arr[k++] = R[j++];
    }
}

void mergeSort(vector<long long>& arr, int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

long long inversionCount(vector<long long>& arr) {
    ans = 0;
    mergeSort(arr, 0, arr.size() - 1);
    return ans;
}

void printArray(const vector<long long>& arr) {
    for (long long num : arr) {
        cout << num << " ";
    }
    cout << endl;
}
```

Step-by-Step Merge and Inversion Tracking

Step	Subarrays (Left - Right)	Comparison	Inversion Count	Merged Result
1	[2] and [3]	2 ≤ 3	0	[2, 3]
2	[2, 3] and [8]	All in order	0	[2, 3, 8]
3	[6] and [1]	6 > 1	1	[1, 6]
4	[2, 3, 8] and [1, 6]	2 > 1	3 (2,3,8 > 1)	
		2 < 6	0	
		3 < 6	0	
		8 > 6	1	[1, 2, 3, 6, 8]

✓ Summary

Merge Step	Inversions Found
[2] and [3]	0
[2, 3] and [8]	0
[6] and [1]	1
[2, 3, 8] and [1, 6]	3 + 1 = 4
Total Inversions	**5**

```
int main() {
    vector<long long> arr = {2, 3, 8, 6, 1};

    cout << "Given Array:" << endl;
    printArray(arr);

    long long inversionCountValue =
inversionCount(arr);

    cout << "Number of inversions: " <<
inversionCountValue << endl;

    return 0;
}
```

```
Given Array:
2 3 8 6 1
Number of inversions: 5
```