Lab-4 106119100 Rajnessh Pandey

1.Given two sorted arrays nums1and nums2 of size m and n respectively, return the median of the two sorted arrays.

Follow up: The overall run time complexity should be O (log (m+n)).

Example 1: Input: nums1 = [1,3], nums2 = [2]

Output:2.00000

Explanation: merged array = [1,2,3] and median is 2.

Example 2: Input: nums1 = [1,2], nums2 = [3,4]

Output:2.50000

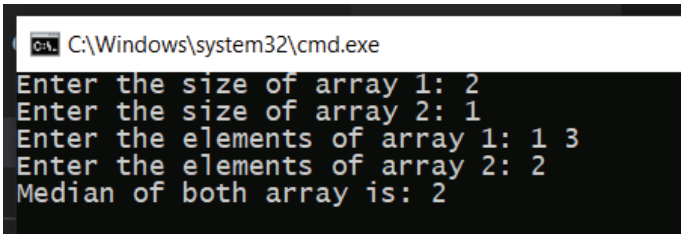
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

Code:

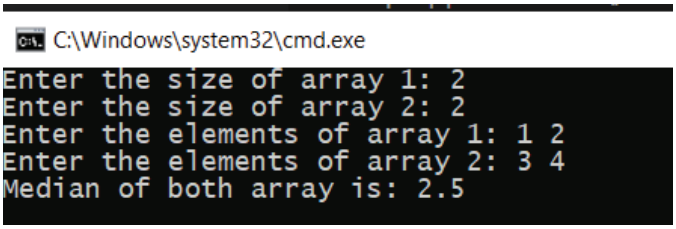
```
#include <bits/stdc++.h>
#define INF 1e7
using namespace std;

int main() {
    int m, n;
    cout << "Enter the size of array 1: ";
    cin >> m;
    cout << "Enter the size of array 2: ";
    cin >> n;
    int a[m], b[n];
    cout << "Enter the elements of array 1: ";
    for(int i = 0; i < m; i++)
        cin >> a[i];
    cout << "Enter the elements of array 2: ";
    for(int i = 0; i < n; i++)
        cin >> b[i];
    cout<<"Median of both array is: ";
    if(m>=n){
        int lo = 0, hi = n * 2;
        while (lo <= hi) {
            int mid2 = (lo + hi) / 2;
            int mid1 = n + m - mid2;
            double w = (mid1 == 0) ? -INF : a[(mid1-1)/2];
            double x = (mid2 == 0) ? -INF : b[(mid2-1)/2];
            double y = (mid1 == m * 2) ? INF : a[(mid1)/2];
            double z = (mid2 == n * 2) ? INF : b[(mid2)/2];
```

```
            if (w > z) lo = mid2 + 1;
            else if (x > y) hi = mid2 - 1;
            else {
                cout<< (max(w,x) + min(z, y)) / 2 << endl;
                return 0;
            }

        }
    }
    else{
        int lo = 0, hi = m * 2;
        while (lo <= hi) {
            int mid2 = (lo + hi) / 2;
            int mid1 = n + m - mid2;

            double w = (mid1 == 0) ? -INF : b[(mid1-1)/2];
            double x = (mid2 == 0) ? -INF : a[(mid2-1)/2];
            double y = (mid1 == n * 2) ? INF : b[(mid1)/2];
            double z = (mid2 == m * 2) ? INF : a[(mid2)/2];

            if (w > z) lo = mid2 + 1;
            else if (x > y) hi = mid2 - 1;
            else {
                cout<< (max(w,x) + min(z, y)) / 2 << endl;
                return 0;
            }
        }
    }
    return 0;
}
```

Output1:



Output2:



2. You are given an integer array nums and you have to return a new counts array. The countsarray has the property where counts[i] is the number of smaller elements to the right of nums[i].

Example 1: Input: nums = [5,2,6,1] Output: [2,1,1,0]

Explanation: To the right of 5 there are 2smaller elements (2 and 1). To the right of 2 there is only 1smaller element (1). To the right of 6 there is 1smaller element (1). To the right of 1 there is 0smaller element.

Code:

```cpp

#include <bits/stdc++.h>

#define val(x) x.first
#define pos(x) x.second

using namespace std;

void merge(vector<int> &count, vector<pair<int, int> > &arr, int left, int mid, int right) {
    vector<pair<int, int> > temp(right-left+1);
    int i = left;
    int j = mid+1;
    int k = 0;

    while (i <= mid && j <= right) {
        if (val(arr[i]) <= val(arr[j])) {
            temp[k++] = arr[j++];
        }
        else {
            count[pos(arr[i])] += right - j + 1;
            temp[k++] = arr[i++];
        }
    }

    while (i <= mid)
        temp[k++] = arr[i++];

    while (j <= right)
        temp[k++] = arr[j++];

    for (int i = left; i <= right; i++)
        arr[i] = temp[i-left];
}

void merge_sort(vector<int> &count, vector<pair<int, int> > &arr, int left, int right) {
    if(left < right) {
        int mid = left + (right-left)/2;
        merge_sort(count, arr, left, mid);
        merge_sort(count, arr, mid+1, right);
        merge(count, arr, left, mid, right);
    }
}

```

```cpp

    while (j <= right)
        temp[k++] = arr[j++];

    for (int i = left; i <= right; i++)
        arr[i] = temp[i-left];
}

void merge_sort(vector<int> &count, vector<pair<int, int> > &arr, int left, int right) {
    if(left < right) {
        int mid = left + (right-left)/2;
        merge_sort(count, arr, left, mid);
        merge_sort(count, arr, mid+1, right);
        merge(count, arr, left, mid, right);
    }
}

void solve(vector<int>& nums) {
    int n = nums.size();
    vector<int> count(n, 0);
    vector<pair<int, int> > arr(n);
    for (int i = 0; i < n; i++)
        arr[i] = make_pair(nums[i], i);

    merge_sort(count, arr, 0, n-1);

    for(int i = 0; i < n; i++)
        cout << count[i] << " ";
    cout << "\n";
}

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter the elements: ";
    for(int i = 0; i < n; i++)
        cin >> nums[i];
    solve(nums);
    return 0;
}

```