Strivers-A2Z-DSA-Sheet-main\02.Binary Search\1D Arrays\03.Implement_lower_upper_bound.cpp

```cpp
/*
QUESTION:
Given an unsorted array Arr[] of N integers and an integer X, find floor and ceiling of X in Arr[0..N-1].

Floor of X is the largest element which is smaller than or equal to X. Floor of X doesn’t exist if X is smaller than the smallest element of Arr[].

Ceil of X is the smallest element which is greater than or equal to X. Ceil of X doesn’t exist if X is greater than the greatest element of Arr[].

Example:

Input:
N = 8, X = 7
Arr[] = {5, 6, 8, 9, 6, 5, 5, 6}
Output: 6 8
Explanation:
Floor of 7 is 6 and ceil of 7 is 8.

APPROACH:
1. Sort the array in ascending order.
2. Use binary search to find the floor and ceil values.
3. The floor value is the largest element smaller than or equal to X, and the ceil value is the smallest element greater than or equal to X.
4. If the floor or ceil values are not found, set them to -1.

CODE:
*/

int lowerbound(int arr[], int n, int x){
    int low = 0, high = n-1;
    int ans = -1;
    while(low<=high){
        int mid = low+(high-low)/2;
        if(arr[mid]<=x){
            ans = mid;
            low = mid+1;
        }
        else{
            high = mid-1;
        }
    }
    if(ans!=-1) ans = arr[ans];
    return ans;
}

int upperbound(int arr[], int n, int x){
    int low = 0, high = n-1;
    int ans = -1;
    while(low<=high){
        int mid = low+(high-low)/2;
```

```cpp
        if(arr[mid]>=x){
            ans = mid;
            high = mid-1;
        }
        else{
            low = mid+1;
        }
    }
    if(ans!=-1) ans = arr[ans];
    return ans;
}

pair<int, int> getFloorAndCeil(int arr[], int n, int x) {
    sort(arr,arr+n);
    int Floor = lowerbound(arr,n,x);
    int Ceil = upperbound(arr,n,x);
    return {Floor,Ceil};
}

// TIME COMPLEXITY: O(NlogN)
// SPACE COMPLEXITY: O(1)

```