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

```cpp
/*
QUESTION:
Given a sorted array arr[] of size N without duplicates, and given a value x. Floor of x is defined as the largest element K in arr[] such that K is smaller than or equal to x. Find the index of K (0-based indexing).

Example 1:
Input:
N = 7, x = 0
arr[] = {1,2,8,10,11,12,19}
Output: -1
Explanation: No element less than 0 is found. So the output is "-1".

Example 2:
Input:
N = 7, x = 5
arr[] = {1,2,8,10,11,12,19}
Output: 1
Explanation: Largest number less than 5 is 2 (i.e K = 2), whose index is 1 (0-based indexing).

APPROACH:
- Initialize low as 0 and high as N-1.
- Iterate using a while loop until low is less than or equal to high.
- Calculate the mid index using mid = low + (high - low) / 2.
- Check if the element at mid index is less than or equal to x.
  - If true, update the answer as mid and move the low pointer to mid+1 to search for a larger element.
  - If false, update the high pointer to mid-1 to search in the lower half of the array.
- Finally, return the answer.

CODE:
*/

int findFloor(vector<long long> v, long long n, long long x) {
    long long low = 0, high = n - 1;
    int ans = -1;
    while (low <= high) {
        long long mid = low + (high - low) / 2;
        if (v[mid] <= x) {
            ans = mid;
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return ans;
}

// TIME COMPLEXITY: O(log N)
// SPACE COMPLEXITY: O(1)

```