Q3

First, sort the array by carrying out advanced sorting algorithms(i.e. The algorithms which takes O(n*logn) to sort an integer list, like merge sort or quick sort etc). Allocate an array of size of n integers.then for every pairs of integers using the variation of binary search(find the position a of first element smaller or equals to the given number, find the position b of last element greater or equals to the given number) to obtain the elements within the range(b - a + 1), and store this number into the array in the form of(result[n] = the number of elements of array A within the range the tuple n).

Time complexity : n*log(n)(sort the array) +
n * (2 * log n), therefore the overall time complexity is n * log(n).

```
binary _search_last_upper_bound(A,n,U)
    ans = -1
    low = 1
    high = n
    while (low <= high) do
        int mid = low + (high - low + 1) / 2
        int midVal = a[mid]
        if (midVal <= key)
            ans = mid
            low = mid + 1
        else if (midVal > key)
            high = mid - 1
    end loop
    return ans;
```

```
binary _search_first_lower_bound(A,n,L)
ans = -1
    low = 1
    high = n
    while (low <= high) do
        int mid = low + (high - low + 1) / 2
        int midVal = a[mid]
        if (midVal <= key)
            low = mid + 1
        else if (midVal > key)
            ans = mid
            high = mid - 1
    end loop
    return ans;
```