Module 04: Assignment 01 (Theory)

Question 1: Write a C++ program that takes N integer numbers and sorts them in non-increasing order using **Merge Sort.**

Answer: Complexity : O(n log n)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 7;
int nums[N];

void merge(int l, int r, int mid)
{
    int left_size = mid - l + 1;
    int L[left_size + 1];

    int right_size = r - mid;
    int R[right_size + 1];
    for (int i = l, j = 0; i <= mid; i++, j++)
    {
        L[j] = nums[i];
    }
    for (int i = mid + 1, j = 0; i <= r; i++, j++)
    {
        R[j] = nums[i];
    }
    L[left_size] = INT_MIN;
    R[right_size] = INT_MIN;

    int lp = 0, rp = 0;
    for (int i = l; i <= r; i++)
    {
        if (L[lp] >= R[rp])
        {
            nums[i] = L[lp];
            lp++;
        }
        else
```

```
        {
            nums[i] = R[rp];
            rp++;
        }
    }
}
void mergesort(int l, int r)
{
    if (l == r)
        return;
    int mid = (l + r) / 2;
    mergesort(l, mid);
    mergesort(mid + 1, r);
    merge(l, r, mid);
}

int main()
{
    int n;
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        cin >> nums[i];
    }

    mergesort(0, n - 1);

    for (int i = 0; i < n; i++)
    {
        cout << nums[i] << " ";
    }
    return 0;
}
```

Question 2: Write a C++ program that takes N integer numbers that are sorted and distinct. The next line will contain an integer k. You need to tell whether K exists in that array or not. If it exists, print its index otherwise print “Not Found”.

You must solve this in O(logn) complexity.

Answer 2: **Binary Search**

```
#include <bits/stdc++.h>
using namespace std;
void binarySearch(vector<int> &nums, int val)
{
    int l = 0, h = nums.size() - 1;
    int mid, flag = 0;
    while (l <= h)
    {
        mid = (l + h) / 2;
        if (nums[mid] == val)
        {
            flag = mid;
            break;
        }
        else if (nums[mid] < val)
        {
            l = mid + 1;
        }
        else
        {
            h = mid - 1;
        }
    }
    (flag) ? cout << flag : cout << "Not Found";
    cout << endl;
}
int main()
{
    vector<int> nums;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        nums.push_back(x);
    }
    int k;
```

```
    cin >> k;
    binarySearch(nums, k);
    return 0;
}
```

Question 3: You are given an array of N positive integers. The next line will contain an integer K. You need to tell whether there exists more than one occurrence of K in that array or not. If there exists more than one occurrence of K print YES, Otherwise print NO.

See the sample input-output for more clarification.

The given array will be sorted in increasing order. And it is guaranteed that at least one occurrence of K will exist. **You must solve this in O(logn) complexity.**

Answer 3: Check duplicate use Binary search algorithm

```
#include <bits/stdc++.h>
using namespace std;
int Bsearch(vector<int> &v, int k)
{
    int l = 0, h = v.size() - 1;
    int mid;
    while (l < h)
    {
        mid = (l + h) / 2;
        if (v[mid] == v[mid + 1] && v[mid] == k)
        {
            return true;
        }
        else if (v[mid] < k)
        {
            l = mid + 1;
        }
        else
        {
            h = mid - 1;
        }
    }
    return false;
```

```
}
int main()
{
    vector<int> v;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        v.push_back(x);
    }
    int k;
    cin >> k;
    if (Bsearch(v, k))
        cout << "YES";
    else
        cout << "NO";
    return 0;
}
```

Question 4: Calculate the time complexity of the following code snippets.

(a)

```
int count = 0;
for (int i = n; i > 0; i /= 2)
{
    for (int j = 0; j < n; j+=5)
    {
        count += 1;
    }
}
```

Answer 4(a): Time Complexity: O(n * log n)

(b)

```
for(int i =1; i*i<n; i++)
{
```

```
    cout << “hello”;
}
```

Answer 4(b): Time Complexity: O(sqrt(n))

(c)

```
for(int i =1; i<n; i=i*2)
{
   for(int j=1; j*j<n; j+=2)
  {
     cout << “hello”;
   }
}
```

Answer 4(c): Time Complexity: O(log n * sqrt(n))

(d)

```
int m = 1;
for(int i=0; m<=n; i++)
{
   m+=i;
}
```

Answer 4(d): Time Complexity: O(sqrt(n))

Question 5: You are given two sorted arrays arr1 and arr2 in descending order. Your task is to merge these two arrays into a new array result using the merge sort technique, but Instead of merging the arrays in ascending order, you need to merge them in descending order to create the result array.

Answer 5: Apply Merge Technique

```
#include <bits/stdc++.h>
using namespace std;
vector<int> merge(vector<int> &arr1, vector<int> &arr2)
{
   vector<int> mergeV;
   int i = 0, j = 0;
   while (i < arr1.size() && j < arr2.size())
   {
```