# Answer Script

## Question No. 1

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

**You can't use any built-in function for sorting.**

**Marks**: 20

| Sample Input | Sample Output |
|---|---|
| 7<br>1 2 9 4 0 2 5 | 9 5 4 2 2 1 0 |
| 6<br>5 3 -1 3 3 8 | 8 5 3 3 3 -1 |

## Answer No. 1

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 7;
int arr[N];
void merge(int l, int r, int mid)
{
    int leftSz = mid - l + 1;
    int lArr[leftSz + 1];

    int rightSz = r - mid; // sz=r-(mid+1)+1
```

```
    int rArr[rightSz + 1];

    // left array
    for (int i = l, j = 0; i <= mid; j++, i++)
    {
        lArr[j] = arr[i];
    }
    // right array
    for (int i = mid + 1, j = 0; i <= r; j++, i++)
    {
        rArr[j] = arr[i];
    }

    lArr[leftSz] = INT_MIN;  // negative infinity
    rArr[rightSz] = INT_MIN; // negative infinity

    // merging
    int lp = 0, rp = 0; // left pointer,right pointer=0
    for (int i = l; i <= r; i++)
    {
        if (lArr[lp] >= rArr[rp])
        {
            arr[i] = lArr[lp];
            lp++;
        }
        else
        {
            arr[i] = rArr[rp];
            rp++;
        }
    }
```

```
}
void mergeSort(int l, int r)
{
    if (l == r)
        return;
    int mid = (l + r) / 2; // mid value
    mergeSort(l, mid);     // left ajaw
    mergeSort(mid + 1, r); // right a jaw
    merge(l, r, mid);      // merge
}
int main()
{
    int sz;
    cin >> sz;
    for (int i = 0; i < sz; i++)
    {
        cin >> arr[i];
    }
    mergeSort(0, sz - 1);
    for (int i = 0; i < sz; i++)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Question No. 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.**

**Marks**: 20

| Sample Input | Sample Output |
|---|---|
| 8<br>-4 0 2 6 9 10 29 54<br>29 | 6 |
| 10<br>0 1 2 3 4 5 6 7 8 9<br>-3 | Not Found |

Answer No. 2

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int sz;
    cin >> sz;
    vector<int> v;
    for (int i = 0; i < sz; i++)
    {
        int x;
```

```
        cin >> x;
        v.push_back(x);
    }
    int data;
    cin >> data;
    int l = 0, r = sz - 1, mid;
    bool flag = true;
    while (l <= r)
    {
        mid = (l + r) / 2;
        if (data == v[mid])
        {
            cout << mid << endl;
            flag = false;
            break;
        }
        else if (data < v[mid])
        {
            r = mid - 1;
        }
        else if (data > v[mid])
        {
            l = mid + 1;
        }
    }
    if (flag)
    {
        cout << "Not Found" << endl;
    }
```

```
    return 0;
}
```

## Question No. 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.**

**Marks**: 20

| Sample Input | Sample Output |
|---|---|
| 7<br>1 3 4 6 6 9 17<br>6 | YES |
| 10<br>0 1 2 3 4 5 6 7 8 9<br>3 | NO |

## Answer No. 3

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int main()
{
    int sz;
    cin >> sz;
    vector<int> v;
    for (int i = 0; i < sz; i++)
    {
        int x;
        cin >> x;
        v.push_back(x);
    }
    int data;
    cin >> data;
    int l = 0, r = sz - 1, mid, indx = 0;
    bool flag = false;
    while (l <= r)
    {
        int mid = (l + r) / 2;
        if (data == v[mid])
        {
            flag = true;
            indx = mid;
            break;
        }
        else if (data < v[mid])
        {
            r = mid - 1;
        }
```

```
        else if (data > v[mid])
        {
            l = mid + 1;
        }
    }
    if (flag && data == v[indx + 1])
    {
        cout << "YES" << endl;
    }
    else
    {
        cout << "NO" << endl;
    }

    return 0;
}
```

| Question No. 4 |
| --- |
| Calculate the time complexity of the following code snippets. |

**Marks**: 20

(a)

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

(b)

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

(c)

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

(d)

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

Answer No. 4

**a) nlog(n)**
**b) sqrt(n)**
**c) log(n)*sqrt(n)**
**d) sqrt(n)**

Question No. 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.

**You cannot use stl sort function here**

**Marks**: 20

| Sample Input | Sample Output |
|---|---|
| 4<br>8 6 4 2<br>4<br>7 5 3 1 | 8 7 6 5 4 3 2 1 |

Answer No. 5

```
#include <bits/stdc++.h>
using namespace std;
void mergeList(vector<int> v1, vector<int> v2, vector<int> vM)
{
    int i = 0, j = 0, k = 0;
    while (i < v1.size() && j < v2.size())
    {
        if (v1[i] ≥ v2[j])
```

```
        {
            vM[k] = v1[i];
            i++;
        }
        else if (v1[i] < v2[j])
        {
            vM[k] = v2[j];
            j++;
        }
        k++;
    }
    // remaining element
    // for array 1
    while (i < v1.size())
    {
        vM[k] = v1[i];
        i++;
        k++;
    }
    // for array 2
    while (j < v2.size())
    {
        vM[k] = v2[j];
        j++;
        k++;
    }
    for (int p = 0; p < (v1.size() + v2.size()); p++)
    {
        cout << vM[p] << " ";
    }
}
```

```cpp
int main()
{
    int n, m;
    cin >> n;
    vector<int> v1, v2;
    for (int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        v1.push_back(x);
    }
    cin >> m;
    for (int i = 0; i < m; i++)
    {
        int x;
        cin >> x;
        v2.push_back(x);
    }
    vector<int> vM(n + m);
    mergeList(v1, v2, vM);

    return 0;
}
```