Answer Script

Question No. 01

WAP that takes n integer numbers, sorts them in non-increasing order using Quick sort.

Sample input	Sample output
5 6 2 3 3 5	6 5 3 3 2
6 5 6 7 8 0 1	8 7 6 5 1 0

Answer No. 01

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

vector<int> quick_sort(vector<int>&a)
{
        if(a.size() <= 1)
        return a;
        int pivot = a.size() - 1;

        vector<int>b, c;
        for(int i = 0; i < a.size(); i++) {
        if(i == pivot)
        continue;
        if(a[i] >= a[pivot])
        b.push_back(a[i]);
        else
        c.push_back(a[i]);
        }
        vector<int>sorted_b = quick_sort(b);
        vector<int>sorted_c = quick_sort(c);
        vector<int>sorted_a;
        for(int i = 0; i < sorted_b.size(); i++)
        sorted_a.push_back(sorted_b[i]);

        sorted_a.push_back(a[pivot]);

        for(int i = 0; i < sorted_c.size(); i++)
        sorted_a.push_back(sorted_c[i]);
        return sorted_a;
}
int main()
{
        int n;
        cin >> n;
        int a;
        vector<int> v;
        for(int i = 0; i < n; i++) {
        cin >> a;
        v.push_back(a);
        }
        vector<int>sorted_a = quick_sort(v);
        for(int i = 0; i < sorted_a.size(); i++) {
        cout << sorted_a[i] << " ";
        }
        return 0;
}
```

Question No. 02

WAP that takes n-1 integer numbers, which contains distinct integers from 1 to n. Exactly one number between 1 to n is missing. Find that number in O(n).

Sample input	**Sample output**
5 1 2 5 46	3
6 1 6 4 3 2	5

Answer No. 02

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int i, num, input, sum1 = 0, sum2 = 0, missing;
   cin >> num;
   for(i = 1; i < num; i++) {
         cin >> input;
         sum1 += input;
   }
```

```
    for(i = 1; i <= num; i++) {
            sum2 += i;
    }
    cout << sum2-sum1 << "\n";
    return 0;
}
```

Question No. 03

WAP that takes n integer numbers and an integer k, and how many pairs of numbers in the array which sums to k. You have to do it inside the Merge Sort function, divide and conquer fashion in O(nlogn).

Sample input	**Sample output**

5
6 1 3 2 4
5
	2

6
5 6 7 8 0 1
16
	0

In sample 1, k = 5. a[1] + a[4] = 1 + 4 = 5 and a[2] + a[3] = 3 + 2 = 5. So the output is 2.
In sample 2, k = 16 and no pair of numbers sum to 16. It is not allowed to use the same index twice. For example a[3] + a[3] = 8 + 8 = 16 but we cannot use index 3 twice.

Answer No. 03

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

int mergeSort(int arr[], int temp[], int left, int right, int k)
{
        int i, j, mid, counter = 0;
        if (right > left)
        {
        mid = (right + left) / 2;

        counter = mergeSort(arr, temp, left, mid, k);
        counter += mergeSort(arr, temp, mid + 1, right, k);
```

```
i = left;
j = mid + 1;
int index = left;
while (i <= mid && j <= right)
{
if (arr[i] <= arr[j])
{
        temp[index++] = arr[i++];
}
else
{
        temp[index++] = arr[j++];
        counter += (mid - i + 1);
}
}
while (i <= mid)
{
temp[index++] = arr[i++];
}
while (j <= right)
{
temp[index++] = arr[j++];
}
for (int i = left; i <= right; i++)
{
arr[i] = temp[i];
}
}

int count = 0;
i = 0, j = right;
while (i < j)
{
if (arr[i] + arr[j] == k)
{
count++;
i++;
j--;
}
else if (arr[i] + arr[j] < k)
{
i++;
}
```

```
        else
        {
        j--;
        }
        }
        return count;
}

int main()
{
        int n;
        cin >> n;
        int arr[n];
        for (int i = 0; i < n; i++)
        {
        cin >> arr[i];
        }
        int k;
        cin >> k;
        int temp[n];
        cout << mergeSort(arr, temp, 0, n - 1, k) << "\n";
        return 0;
}
```

Question No. 04

15

WAP that takes 2 integer arrays with distinct elements as input, and checks if array 1 is a subset of array 2. Solve the problem in O(nlogn) or better.

Sample input	**Sample output**

3
7 2 3
5
7 6 3 2 1
	YES

3
1 2 3
3
3 2 1
	YES

3
1 2 4
3
3 2 1
	NO

In Sample 3, array 1 is not a subset of array 2 because 4 does not occur in array 2.

Answer No. 04

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

int main() {
        int n1, n2;
        cin >> n1;
        int a1[n1];
        for (int i = 0; i < n1; i++) {
        cin >> a1[i];
        }
        cin >> n2;
        int a2[n2];
        for (int i = 0; i < n2; i++) {
        cin >> a2[i];
        }
        sort(a1, a1 + n1);
        sort(a2, a2 + n2);
        int i = 0, j = 0;
        while (i < n1 && j < n2) {
        if (a1[i] < a2[j]) {
        cout << "NO" << endl;
        return 0;
        }
        else if (a1[i] == a2[j]) {
        i++;
        }
        j++;
        }
        if (i == n1) {
        cout << "YES" << "\n";
        }
        else {
        cout << "NO" << "\n";
        }
        return 0;
}
```

Question No. 05

Take the linked-list class that we created in our Week 3 Lab Lecture. Add the following functions to the linked list.

- **int getSize()** -> This function will return the number of elements in the linked-list. This function should work in O(1). For this keep track of a size variable and update it when we insert a new value in the linked-list.
- **int getValue(index)** -> This function will return the value present in the input index. If the index is greater or equal to the size of the linked-list return -1.
- **void printReverse()** -> This function will print the linked list in reverse order. You don't need to reverse the linked list. Just need to print it in reverse order. **You need to do this recursively**. You cannot just take the elements in an array or vector and then print them in reverse order.
- **void swapFirst() ->** This function will swap the first two nodes in the linked list. If the linked-list contains less than 2 elements then just do nothing and return.

To check your code add the following code in your main function.

```
LinkedList l;
cout<<l.getSize()<<"\n";
l.InsertAtHead(5);
cout<<l.getSize()<<"\n";
l.InsertAtHead(6);
l.InsertAtHead(30);
cout<<l.getSize()<<"\n";
l.InsertAtHead(20);
l.InsertAtHead(30);

cout<<l.getValue(2)<<"\n";

cout<<l.getValue(6)<<"\n";

l.printReverse();
l.Traverse();
l.swapFirst();
```

```
    l.Traverse();
    l.printReverse();
```

Output

```
0
1
3
30
-1
5 6 30 20 30
30 20 30 6 5
20 30 30 6 5
5 6 30 30 20
```

Answer No. 05

```
#include<bits/stdc++.h>
using namespace std;
class node
{
public:
        int data;
        node* nxt;
};

class LinkedList
{
private:
        node* head;
        int size;
public:
        LinkedList()
        {
        head = NULL;
        size = 0;
        }

        void InsertAtHead(int value)
        {
        node* newNode = new node();
```

```
newNode->data = value;
newNode->nxt = head;
head = newNode;
size++;
}

int getSize()
{
return size;
}

int getValue(int index)
{
if (index >= size) return -1;
node* current = head;
for (int i = 0; i < index; i++)
{
current = current->nxt;
}
return current->data;
}

void printReverse(node* current)
{
if (current == NULL) return;
printReverse(current->nxt);
cout << current->data << " ";
}

void printReverse()
{
printReverse(head);
cout<<endl;
}

void Traverse()
{
node* current = head;
while (current != NULL)
{
cout << current->data << " ";
current = current->nxt;
}
```

```
        cout << endl;
        }

        void swapFirst()
        {
        if (size < 2) return;
        node* firstNode = head;
        node* secondNode = head->nxt;
        firstNode->nxt = secondNode->nxt;
        secondNode->nxt = firstNode;
        head = secondNode;
        }
};

int main()
{
        LinkedList l;
        cout<<l.getSize()<<"\n";
        l.InsertAtHead(5);
        cout<<l.getSize()<<"\n";
        l.InsertAtHead(6);
        l.InsertAtHead(30);
        cout<<l.getSize()<<"\n";
        l.InsertAtHead(20);
        l.InsertAtHead(30);

        cout<<l.getValue(2)<<"\n";

        cout<<l.getValue(6)<<"\n";

        l.printReverse();
        l.Traverse();
        l.swapFirst();
        l.Traverse();
        l.printReverse();

        return 0;
}
```

Question No. 06

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.
Note - The given array will be sorted in increasing order. And it is guaranteed that at least one occurrence of k will exist.

** Solve this problem using binary search means O(logn)**

Sample input 1-	Sample output 1-

7
1 3 4 6 6 9 17
6
	YES

Sample input 2-	Sample output 2-

10
1 2 3 4 5 6 7 8 9 10
5
	NO

Explanation -
In sample input 1 there exist two occurrences of k, hence the answer is YES

Answer No. 06

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

int binarySearchFirst(int a[], int n, int k) {
        int left = 0, right = n-1, result = -1;
        while (left <= right) {
        int mid = left + (right-left)/2;
        if (a[mid] == k) {
        result = mid;
        right = mid-1;
        } else if (a[mid] < k) {
        left = mid+1;
        } else {
        right = mid-1;
        }
        }
        return result;
}
```

```
int binarySearchLast(int a[], int n, int k) {
        int left = 0, right = n-1, result = -1;
        while (left <= right) {
        int mid = left + (right-left)/2;
        if (a[mid] == k) {
        result = mid;
        left = mid+1;
        } else if (a[mid] < k) {
        left = mid+1;
        } else {
        right = mid-1;
        }
        }
        return result;
}

int main() {
        int n;
        cin >> n;
        int a[n];
        for (int i = 0; i < n; i++) {
        cin >> a[i];
        }
        int k;
        cin >> k;
        int first = binarySearchFirst(a, n, k);
        int last = binarySearchLast(a, n, k);
        if (last - first > 0) {
        cout << "YES" << "\n";
        } else {
        cout << "NO" << "\n";
        }
        return 0;
}
```

Question No. 07

You are given an array of n positive integers. Your task is to remove the element from the range a position to b position.
After removing the element print the all elements left.

Sample input 1- Sample output
1-
6
1 4 6 2 8 7 1 8 7
2 4

Explanation -
In simple input 1, we remove the elements from 2 positions to 4 positions.

Answer No. 07

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

int main() {
        int n, a, b;
        cin >> n;
        vector<int> v;
        for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        v.push_back(x);
        }
        cin >> a >> b;
        v.erase(v.begin() + a - 1, v.begin() + b);
        for (int x : v) {
        cout << x << " ";
        }
        return 0;
}
```

Question No. 08

Write a C++ program that returns the elements in a vector that are even numbered.

1)Take 5 elements into a vector.

2)Define a function named even_generator(), which receives a vector and returns a vector that only contains even numbers.

Note: You don't need to take input from the user.

Answer No. 08

```
#include<bits/stdc++.h>
using namespace std;
vector<int> even_generator(vector<int> v) {
    vector<int> even_numbers;
    for (int i = 0; i < v.size(); i++) {
        if (v[i] % 2 == 0) {
            even_numbers.push_back(v[i]);
        }
    }
    return even_numbers;
}
int main() {
    vector<int> v = {8, 11, 7, 4, 5};
    vector<int> even_numbers = even_generator(v);
    cout << "Even numbers in the given vector: ";
    for (int i = 0; i < even_numbers.size(); i++) {
        cout << even_numbers[i] << " ";
    }
    return 0;
}
```