Week 1: Array

Aim:

In a faraway Galaxy of Tilky Way, there was a
planet Tarth where the sport of Competitive Coding was very popular.
According to legends, there lived a setter known for loving knapsack type
problems.

Given N objects in a row, with weights W1,W2,…,WN,
you need to find the maximum number of consecutive objects you can fill in a
bag of maximum capacity C such that the total weight of objects taken is at
least K.

In other words, pick objects such that-The total
weight of collected objects is at least K.

The total weight does not exceed C.

The objects picked must be consecutive (i.e. a
subarray of the objects need to be picked) The number of objects is
maximized. You need to print this maximum value.

Note-If no such object could be picked, then the
answer is obviously 0.

Input

- The first line of input contains T, number of test cases in a file.
- The next line contains three integers, N, C and K, as described in the problem statement.
- The next line contains N space separated integers, denoting Wi, i.e. weight of the object.

Output

- For test case, output the maximum number of objects you can pick

Program:

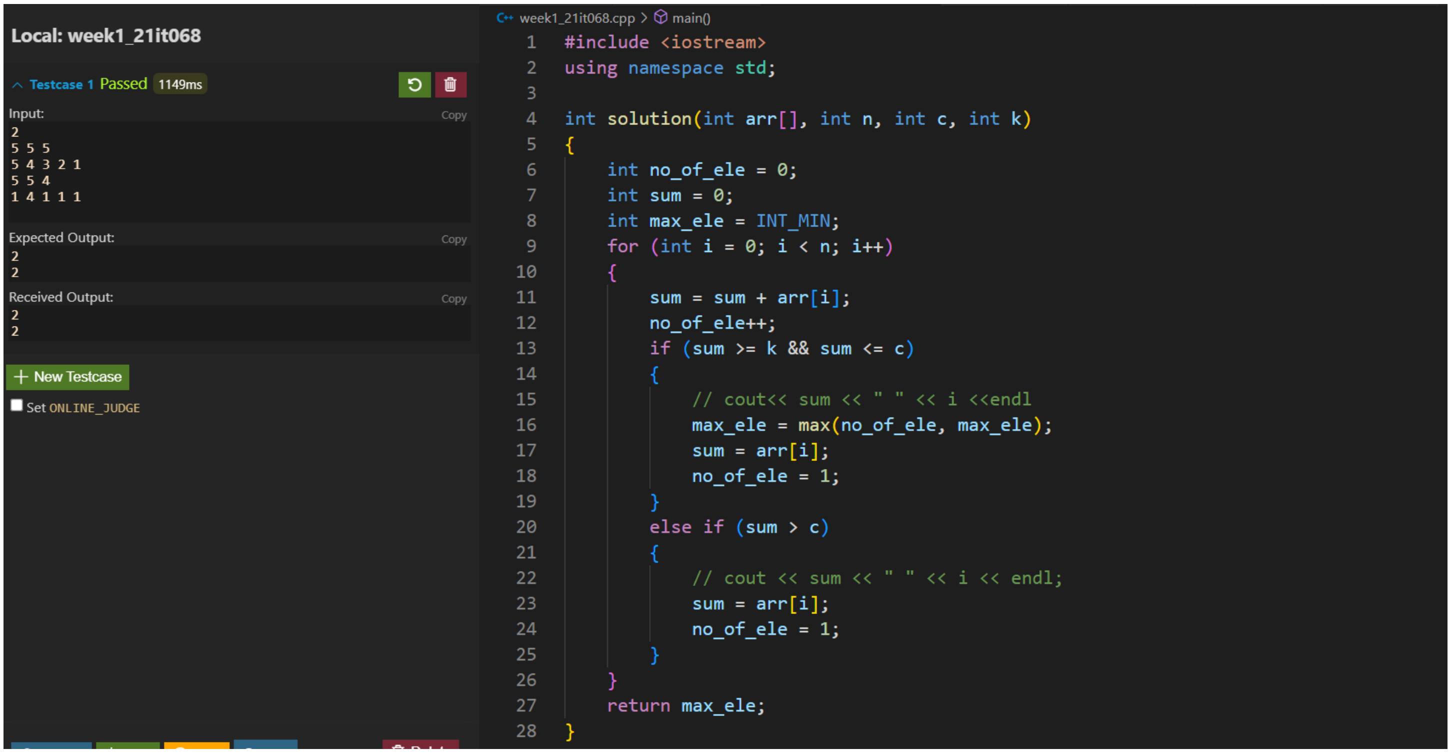
```
#include <iostream>
using namespace std;

int solution(int arr[], int n, int c, int k)
{
    int no_of_ele = 0;
    int sum = 0;
    int max_ele = INT_MIN;
    for (int i = 0; i < n; i++)
    {
        sum = sum + arr[i];
        no_of_ele++;
        if (sum >= k && sum <= c)
        {
            // cout<< sum << " " << i <<endl
            max_ele = max(no_of_ele, max_ele);
            sum = arr[i];
            no_of_ele = 1;
        }
        else if (sum > c)
        {
            // cout << sum << " " << i << endl;
            sum = arr[i];
            no_of_ele = 1;
        }
    }
    return max_ele;
}
```

```
int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int n, c, k;
        cin >> n >> c >> k;
        int arr[n];
        for (int i = 0; i < n; i++)
        {
            cin >> arr[i];
        }
        int ans = solution(arr, n, c, k);
        cout << ans << endl;
    }

    return 0;
}
```

Input & Output:



Local: week1_21it068

Input:

```
2
5 5 5
5 4 3 2 1
5 5 4
1 4 1 1 1
```

Expected Output:

```
2
2
```

Received Output:

```
2
2
```

```cpp
#include <iostream>
using namespace std;

int solution(int arr[], int n, int c, int k)
{
    int no_of_ele = 0;
    int sum = 0;
    int max_ele = INT_MIN;
    for (int i = 0; i < n; i++)
    {
        sum = sum + arr[i];
        no_of_ele++;
        if (sum >= k && sum <= c)
        {
            // cout<< sum << " " << i <<endl
            max_ele = max(no_of_ele, max_ele);
            sum = arr[i];
            no_of_ele = 1;
        }
        else if (sum > c)
        {
            // cout << sum << " " << i << endl;
            sum = arr[i];
            no_of_ele = 1;
        }
    }
    return max_ele;
}
```

Conclusion:

In the given problem we were given an array of weights we needed output of the maximum length of the subarray possible such that the sum of the sub array does not exceed the minimum capacity of the bag and is at least equal to the value K given in the problem.

We created a data structure Array store the data. By using kadane's algorithm we calculated the maximum number of elements having sum greater than k and less than c.

With this practice we have been able to realize the necessity of algorithms and a structure to manipulate the data