Sliding Window Problem

Sliding Window

You are given an array of integers, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

[5, 2, 4, 6, 3, 1]
[5, 2, 4, 6, 3, 1]
[5, 2, 4, 6, 3, 1]
[5, 2, 4, 6, 3, 1]
[5, 2, 4, 6, 3, 1]

k = 2

Sliding Window: Challenge

Write a program in which an array is given as input along with the window size k, check the sum of the k integers of the window size and return the highest sum after checking every consecutive k items in the list.

k = 2

[5, 2, 4, 6, 3, 1]	Sum = 7
[5, 2, 4, 6, 3, 1]	Sum = 6
[5, 2, 4, 6, 3, 1]	Sum = 10
[5, 2, 4, 6, 3, 1]	Sum = 9
[5, 2, 4, 6, 3, 1]	Sum = 4

Sliding Window: Challenge

Write a program in which an array is given as input along with the window size k, check the sum of the k integers of the window size and return the highest sum after checking every consecutive k items in the list.

k = 3

[5, 2, 4, 6, 3, 1]	Sum = 11
[5, 2, 4, 6, 3, 1]	Sum = 12
[5, 2, 4, 6, 3, 1]	Sum = 13
[5, 2, 4, 6, 3, 1]	Sum = 10

Sliding Window: Challenge

Write a program in which an array is given as input along with the window size k, check the sum of the k integers of the window size and return the highest sum after checking every consecutive k items in the list.

k = 4

[5, 2, 4, 6, 3, 1]	Sum = 17
[5, 2, 4, 6, 3, 1]	Sum = 15
[5, 2, 4, 6, 3, 1]	Sum = 14

Solution

Let's make a function first that takes the starting and ending index and calculate the sum of elements between these indexes

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

// Global Array
int arr[100];

// Function definition
int checkLargest(int start, int end)
{
    int sum = 0;
    for (int idx = start; idx < end; idx = idx + 1)
    {
        sum = sum + arr[idx];
    }
    return sum;
}
```

Solution

Now, Let's take the inputs from the user.

```
main()
{
    int arr_length, k, highestSum, sum, s = 0;
    cout << "How many numbers you want to Enter: ";
    cin >> arr_length;
    for (int idx = 0; idx < arr_length; idx = idx + 1)
    {
        cout << "Enter " << idx + 1 << " Element: ";
        cin >> arr[idx];
    }
    cout << "Enter Window Size: ";
    cin >> k;

}
```

Solution

Now, Let's check the sum for the first time.

```
main()
{
    int arr_length, k, highestSum, sum, s = 0;
    cout << "How many numbers you want to Enter: ";
    cin >> arr_length;
    for (int idx = 0; idx < arr_length; idx = idx + 1)
    {
        cout << "Enter " << idx + 1 << " Element: ";
        cin >> arr[idx];
    }
    cout << "Enter Window Size: ";
    cin >> k;
    highestSum = checkLargest(s, k);

}
```

Solution

Now, Let's keep on iterating till the end of the array and then store the largest sum in a separate variable.

```
main()
{
    int arr_length, k, highestSum, sum, s = 0;
    cout << "How many numbers you want to Enter: ";
    cin >> arr_length;
    for (int idx = 0; idx < arr_length; idx = idx + 1)
    {
        cout << "Enter " << idx + 1 << " Element: ";
        cin >> arr[idx];
    }
    cout << "Enter Window Size: ";
    cin >> k;
    highestSum = checkLargest(s, k);
    while (k < arr_length)
    {
        s = s + 1;
        k = k + 1;
        int sum = checkLargest(s, k);
        if (highestSum < sum)
        {
            highestSum = sum;
        }
    }
}
```

Solution

Now, Let's print the final sum.

```
main()
{
    int arr_length, k, highestSum, sum, s = 0;
    cout << "How many numbers you want to Enter: ";
    cin >> arr_length;
    for (int idx = 0; idx < arr_length; idx = idx + 1)
    {
        cout << "Enter " << idx + 1 << " Element: ";
        cin >> arr[idx];
    }
    cout << "Enter Window Size: ";
    cin >> k;
    highestSum = checkLargest(s, k);
    while (k < arr_length)
    {
        s = s + 1;
        k = k + 1;
        int sum = checkLargest(s, k);
        if (highestSum < sum)
        {
            highestSum = sum;
        }
    }
    cout << "Highest Sum = " << highestSum;
}
```

Learning Objective

In this lecture, we learnt how to use arrays and loops to solve real world problems.

Self Assessment

1. Write a C++ program that takes two arrays and returns true if the second array is the same as the first array but shifted to the right by 1 element.

Note

- Both input arrays will be of the same length, and will have a minimum length of 2.
- The values of the 0-indexed element in the second list and the n-1th indexed element in the first list do not matter.

Self Assessment

- Test Cases

Input	Output
First Array: [1, 2] Second Array: [5, 1]	true
First Array: [1, 2] Second Array: [5, 5]	false
First Array: [1, 2, 3, 4, 5] Second Array: [5, 5, 1, 2, 3]	false
First Array: [1, 2, 3, 4, 5] Second Array: [0, 1, 2, 3, 4]	true