



Ahsania Mission University of Science & Technology

Department of Computer Science and Engineering

1st Batch, 2nd Year 2st Semester, Fall 2025

Course Code: CSE 2202

Course Title: Computer Algorithms Sessional

Lab Report

Experiment No :03

Experiment Date :17/05/2025

Submission date:23/05/2025

Submitted To

Md:Fahim Faisal,Lecturer,

Department of Computer Science and Engineering

Faculty of Engineering, Ahsania Mission University of Science & Technology

Submitted By

Name : **MD:Rubyait Roman Rifat.**

ID : **1012320005101014**

Task 01: A C++ program that will find Leaders in an array.

Theory:

In the given problem, we need to identify all the leaders in an array. An element is considered a leader if it is strictly greater than all the elements to its right side, and the rightmost element is always a leader. The goal is to iterate through the array from the rightmost element to the leftmost, keeping track of the maximum element encountered so far. This approach ensures that each leader is identified in a single pass.

Code:

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

void arrayLeader(int arr[] , int Size)
{
    int leaders[Size];
    int l_count = 0;

    int maxRight = arr[Size - 1];
    leaders[l_count++] = maxRight;

    for(int i = Size - 2 ; i >= 0 ; i--)
    {
        if(arr[i] > maxRight)
        {
            maxRight = arr[i];
            leaders[l_count++] = maxRight;
        }
    }

    for(int i = l_count - 1 ; i >= 0 ; i--)
    {
        cout << leaders[i] << " ";
    }
}

int main()
{
    int n;
    cout << "Enter the size of the array: ";
```

```
cin >> n;

int a[n];
cout << "Enter array elements: ";
for(int i = 0 ; i < n ; i++)
{
    cin >> a[i];
}

cout << "Leader of an Array: ";
arrayLeader(a , n);

cout<<"\n";

return 0;
}
```

Output:

```
Enter the size of the array: 6
Enter array elements: 16 17 4 3 5 2
Leader of an Array: 17 5 2

Process returned 0 (0x0)   execution time : 17.432 s
Press any key to continue.
|
```

Task 02: A C++ program that will find the Maximum Subarray Sum.

Theory:

The objective of this task is to find the subarray with the maximum sum within a given array. A subarray is defined as a contiguous sequence of elements within the array. To solve this, we employ Kadane's algorithm, which uses a dynamic programming approach to keep track of the maximum sum subarray ending at each position. The algorithm iteratively updates the global maximum sum as well as the current maximum sum, ensuring that the optimal solution is found.

Code:

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

int subArray(int arr[] , int Size)
{
    int max_global = arr[0];
    int max_current = arr[0];

    for(int i = 1 ; i < Size ; i++)
    {
        max_current = max(arr[i] , max_current + arr[i]);
        if(max_current > max_global)
        {
            max_global = max_current;
        }
    }
    return max_global;
}

int main()
{
    int t;
    cout << "Enter test cases: ";
    cin >> t;

    while(t--)
    {
        int n;
        cout << "Enter the size of the array: ";
```

```

    cin >> n;

    int a[n];
    cout << "Enter array elements: ";
    for(int i = 0 ; i < n ; i++)
    {
        cin >> a[i];
    }

    int max_sum = subArray(a , n);
    cout << max_sum << "\n";
}

return 0;
}

```

Output:

```

Enter test cases: 3
Enter the size of the array: 9
Enter array elements: -2 1 -3 4 -1 2 1 -5 4
6
Enter the size of the array: 1
Enter array elements: 1
1
Enter the size of the array: 5
Enter array elements: 5 4 -1 7 8
23

Process returned 0 (0x0)    execution time : 70.560 s
Press any key to continue.

```