BCSE204P- Design and Analysis of Algorithms Lab
E1-Slot-L43-L44 Lab
In-Lab Practice (IPS) Exercise-3

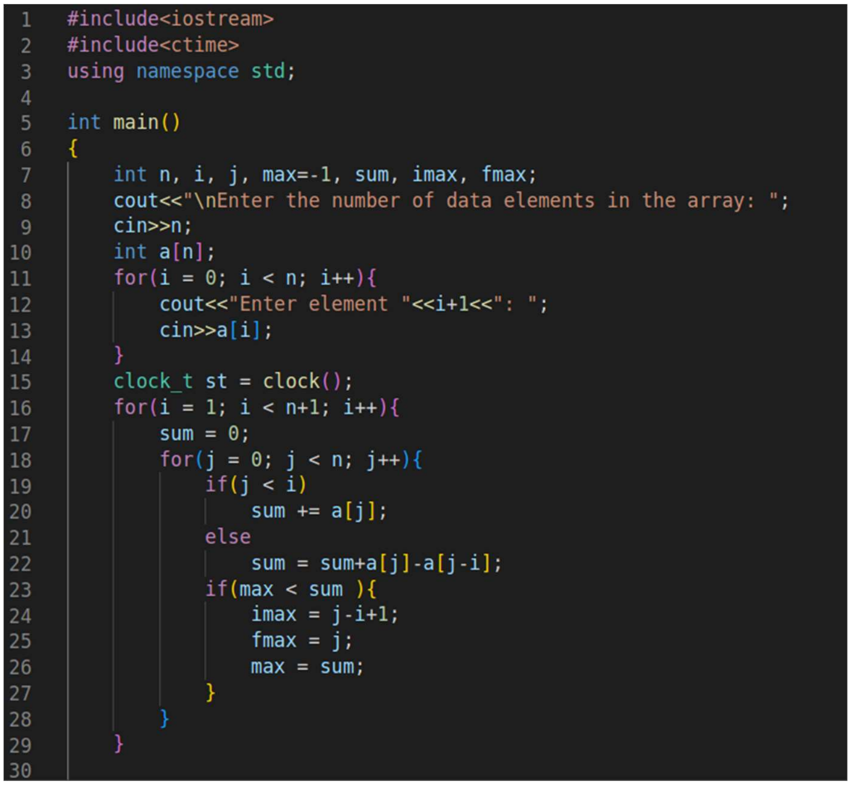
Name: R. Muhmmad Abrar
Register no: 21BRS1713

MAX-SUB ARRAY
Any Two question Output:

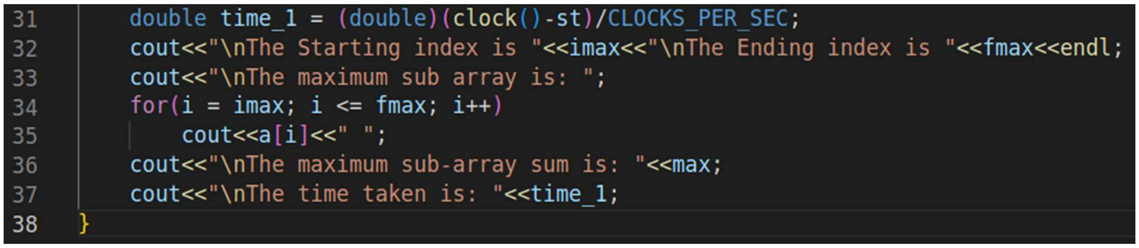
Q1).

Maximum Sum Subarray problem takes an array of positive and negative integers S as input and finds the subarray of S having maximum sum. For example, consider an array with 16 elements, 13, -3, -25, 20, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7. Maximum subarray starts at index 7 and ends at index 10 (programming indices) and sum is 43. Write a C++ code to solve the problem using a naive approach. Print the starting index, end index and sum of the subarray whose sum is maximum. What is the time complexity of the algorithm?

Code:

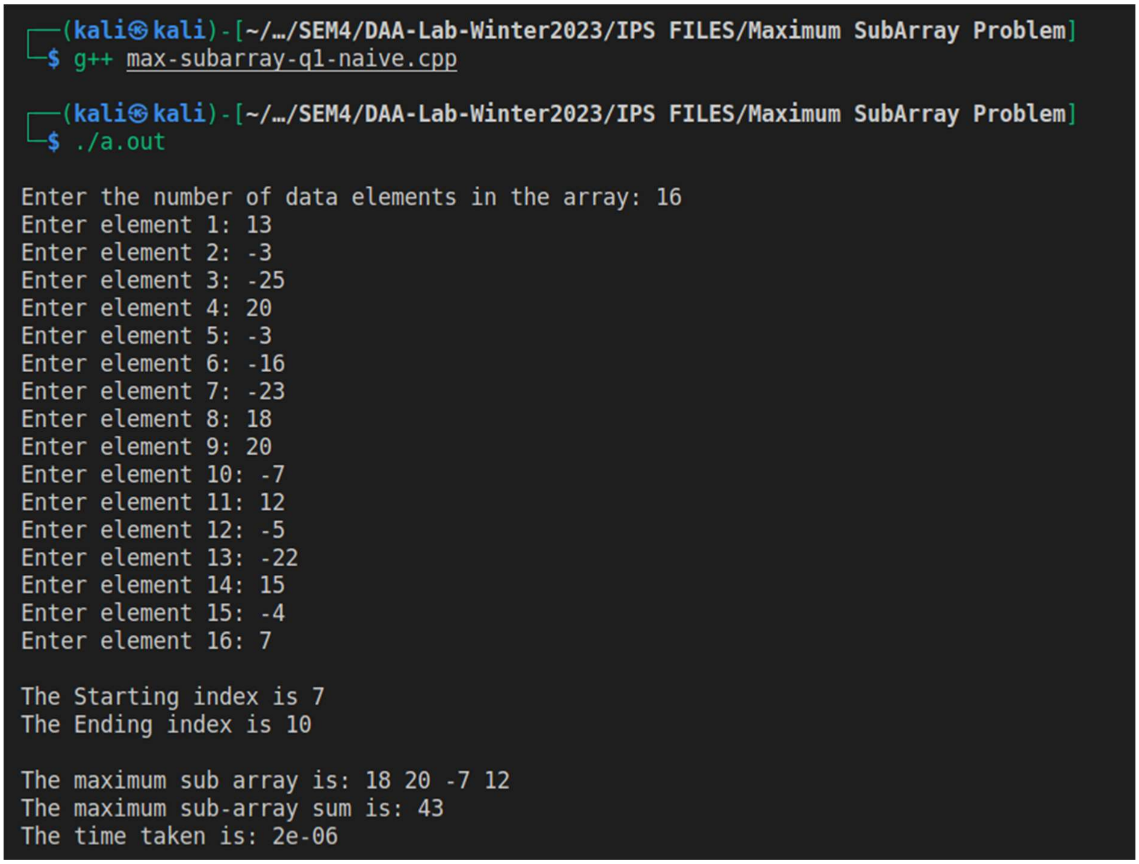


```cpp
#include<iostream>
#include<ctime>
using namespace std;

int main()
{
    int n, i, j, max=-1, sum, imax, fmax;
    cout<<"\nEnter the number of data elements in the array: ";
    cin>>n;
    int a[n];
    for(i = 0; i < n; i++){
        cout<<"Enter element "<<i+1<<": ";
        cin>>a[i];
    }
    clock_t st = clock();
    for(i = 1; i < n+1; i++){
        sum = 0;
        for(j = 0; j < n; j++){
            if(j < i)
                sum += a[j];
            else
                sum = sum+a[j]-a[j-i];
            if(max < sum ){
                imax = j-i+1;
                fmax = j;
                max = sum;
            }
        }
    }

```



```
    double time_1 = (double)(clock()-st)/CLOCKS_PER_SEC;
    cout<<"\nThe Starting index is "<<imax<<"\nThe Ending index is "<<fmax<<endl;
    cout<<"\nThe maximum sub array is: ";
    for(i = imax; i <= fmax; i++)
        cout<<a[i]<<" ";
    cout<<"\nThe maximum sub-array sum is: "<<max;
    cout<<"\nThe time taken is: "<<time_1;
}
```

Output:



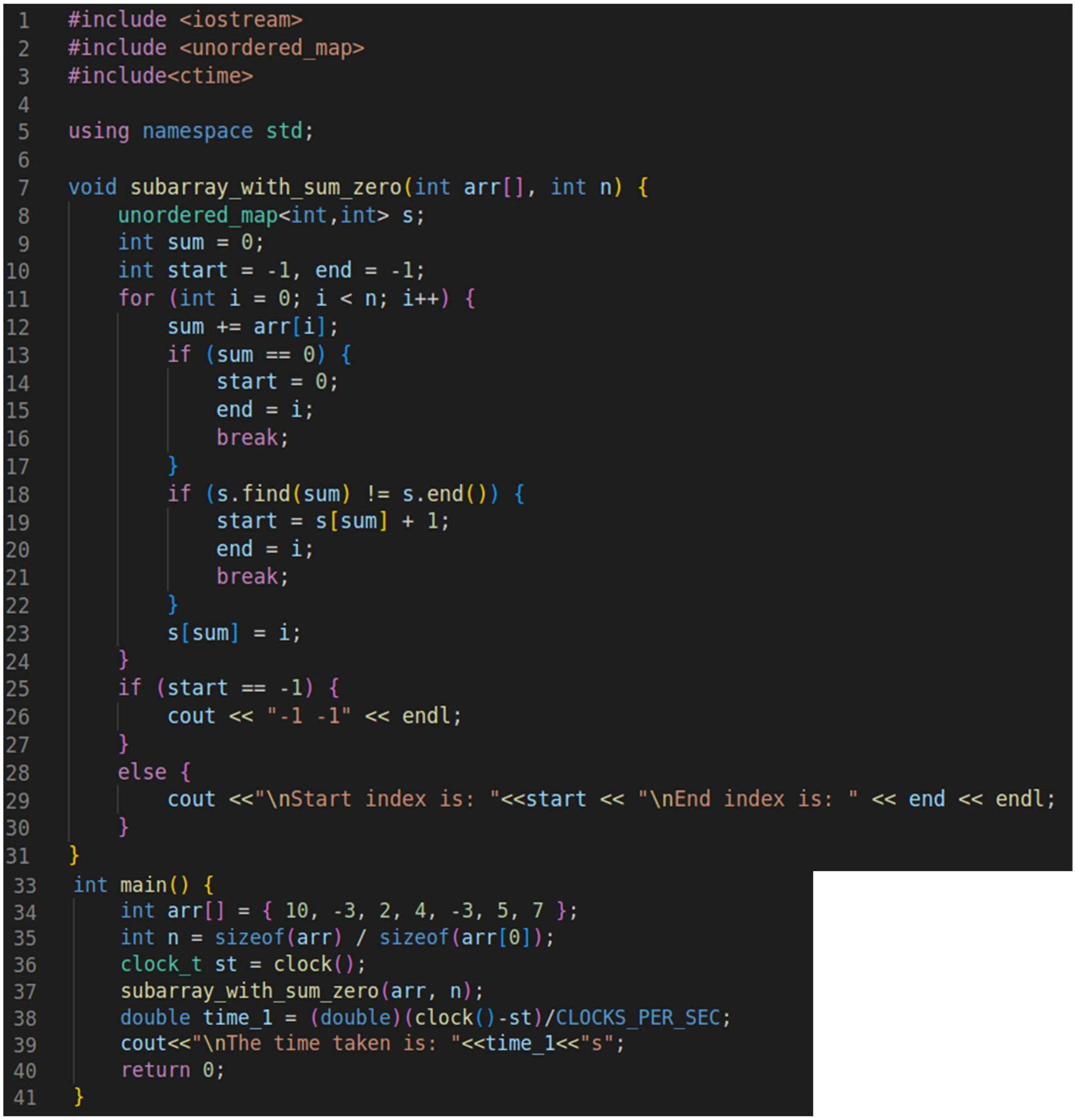
```
┌──(kali㉿kali)-[~/…/SEM4/DAA-Lab-Winter2023/IPS FILES/Maximum SubArray Problem]
└─$ g++ max-subarray-q1-naive.cpp

┌──(kali㉿kali)-[~/…/SEM4/DAA-Lab-Winter2023/IPS FILES/Maximum SubArray Problem]
└─$ ./a.out

Enter the number of data elements in the array: 16
Enter element 1: 13
Enter element 2: -3
Enter element 3: -25
Enter element 4: 20
Enter element 5: -3
Enter element 6: -16
Enter element 7: -23
Enter element 8: 18
Enter element 9: 20
Enter element 10: -7
Enter element 11: 12
Enter element 12: -5
Enter element 13: -22
Enter element 14: 15
Enter element 15: -4
Enter element 16: 7

The Starting index is 7
The Ending index is 10

The maximum sub array is: 18 20 -7 12
The maximum sub-array sum is: 43
The time taken is: 2e-06
```

The time complexity of my algorithm is O(n^2).

Q4).

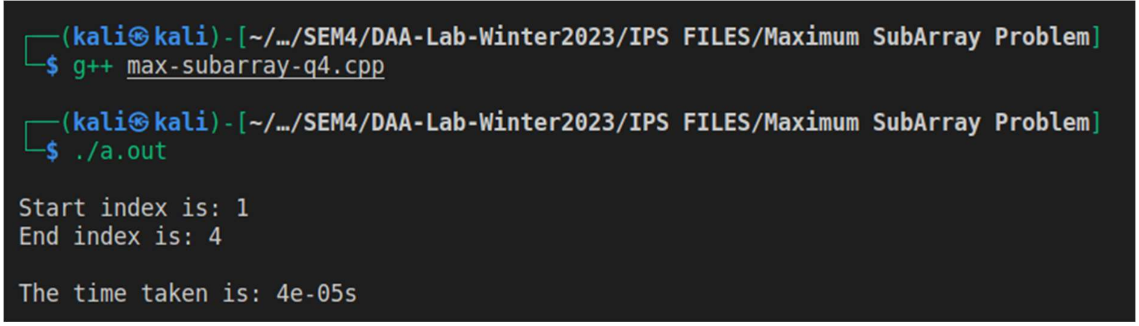
Given an array of positive and negative integers, develop a linear algorithm and write a C++ code to check if there exist a subarray whose sum is zero is present in it. If such a subarray exist then print start and end index of first occurrence. Otherwise print -1 for both start and end index. For example, if the array contains seven elements 10, -3, 2, 4, -3, 5, 7 then print start index as 1 and end index as 4.

Code:



```cpp
#include <iostream>
#include <unordered_map>
#include<ctime>

using namespace std;

void subarray_with_sum_zero(int arr[], int n) {
    unordered_map<int,int> s;
    int sum = 0;
    int start = -1, end = -1;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
        if (sum == 0) {
            start = 0;
            end = i;
            break;
        }
        if (s.find(sum) != s.end()) {
            start = s[sum] + 1;
            end = i;
            break;
        }
        s[sum] = i;
    }
    if (start == -1) {
        cout << "-1 -1" << endl;
    }
    else {
        cout <<"\nStart index is: "<<start << "\nEnd index is: " << end << endl;
    }
}
int main() {
    int arr[] = { 10, -3, 2, 4, -3, 5, 7 };
    int n = sizeof(arr) / sizeof(arr[0]);
    clock_t st = clock();
    subarray_with_sum_zero(arr, n);
    double time_1 = (double)(clock()-st)/CLOCKS_PER_SEC;
    cout<<"\nThe time taken is: "<<time_1<<"s";
    return 0;
}
```

Output:



```
┌──(kali㉿kali)-[~/…/SEM4/DAA-Lab-Winter2023/IPS FILES/Maximum SubArray Problem]
└─$ g++ max-subarray-q4.cpp

┌──(kali㉿kali)-[~/…/SEM4/DAA-Lab-Winter2023/IPS FILES/Maximum SubArray Problem]
└─$ ./a.out

Start index is: 1
End index is: 4

The time taken is: 4e-05s
```