Introduction to Basic Data Structures

Problem-1: Running Sum of an Array

Description:

Given an array **nums** of integers, define a running sum of the array as `runningSum[i] = sum(nums[0]…nums[i])`. Write a C++ program to compute the running sum of the given array and print the result.

Note: Solve this problem using function and Vector.

Input

- The first line contains an integer n representing the size of the array.
- The second line contains n integers representing the elements of the array `nums`.

Output

- Print the running sum of the array as a sequence of integers separated by spaces.

Example:

Input	Output
4 1 2 3 4	1 3 6 10

Explanation:

Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

Problem 2: Equilibrium Index

Description: Given an array of integers, find the equilibrium index. An equilibrium index is an index such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes.

Example:

Input	Output
7 -7 1 5 2 -4 3 0	3

Explanation: At index 3, the sum of elements before it is -1 and after it is also -1

-7	1	5	2	-4	3	0
0	1	2	3	4	5	6

Sum of before index 3 = -7 + 1 + 5 = -1
Sum of after index 3 = -4 + 3 + 0 = -1

Problem-3: Search Query

1. WAP that takes an array of size n and q queries as input. For each query you will be given a number. For each query you have to print 'YES' if the number is present in the array, otherwise print 'No'. Solve the problem in **O((n+q)*logn)**

Sample input	Sample output
5 6 3 2 1 8 4 1 5 2	YES NO YES NO

9	

Explanation: You are given an array of size 5 and 4 queries. In the first query you are given 1. 1 is present in the array so we print ‘YES’. In the second query 5 is not present in the array so we print ‘NO’. Third and Fourth query are similar.

Problem-4: Complexity Analysis

1. Write the time complexity of each of the code segments shown below.
   -

```
int k=1;
while(k<=n){
        cout<<k<<endl;
        k=k*2;
}
```

O(logn)

   -

```
for (int i = 0; i < n; i++)
      for (int j = i; j > 0; j--)
         cout << i << j;
```

O(n*n)

   -

```
for (int i = 0; i < n; i++)
      for (int j = i; j > 0; j--)
         for(int k=j; k > 0; k--)
            cout << i << j << k;
```

O(n*n*n)

   -

```
for(int i=n/2;i<=n;i++){
        for(int j=1;j<=n;j=j*2){
                cout<<i<<j<<endl;
        }
}
```

O(nlogn)

   -

```
for(int i=n/2;i<=n;i++){
        for(int j=1;j<=n;j=j+i){
                cout<<i<<j<<endl;
        }
}
```

O(nlogn)

Problem-5: Summation from 1 to N

Note: Solve this problem in O(1) Complexity.