NAME:SAURABH RAJ
REGISTERED EMAIL :saurabhraj25aug2004@gmail.com
COURSE NAME:DECODE DSA WITH C++
BATCH:DECODE 2.0
MODULE NAME:PREFIX SUM
MOBILE NUMBER:8434283953

QUESTION1:

1. Calculate the **sum** of the elements of `nums` between indices `left` and `right` **inclusive** where `left <= right`.

Implement the `NumArray` class:

- `NumArray(int[] nums)` Initializes the object with the integer array `nums`.
- `int sumRange(int left, int right)` Returns the **sum** of the elements of `nums` between indices `left` and `right` **inclusive** (i.e. `nums[left] + nums[left + 1] + ... + nums[right]`). [Leetcode 303]

Example 1:

Input

["NumArray", "sumRange", "sumRange", "sumRange"]

[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]

Output

[null, 1, -1, -3]

Explanation

NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);

numArray.sumRange(0, 2); // return (-2) + 0 + 3 = 1

numArray.sumRange(2, 5); // return 3 + (-5) + 2 + (-1) = -1

numArray.sumRange(0, 5); // return (-2) + 0 + 3 + (-5) + 2 + (-1) = -3

SOLUTION;

```cpp
//pw
class NumArray {
public:
    vector<int>pre;

    NumArray(vector<int>& nums) {
        pre =vector<int>(nums.size());
        pre[0] =nums[0];
```

```
        int n =nums.size();
        for(int i=1;i<n;i++) pre[i] =pre[i-1] + nums[i];
    }

    int sumRange(int left, int right) {
        if(left==0) return pre[right];
        return pre[right]-pre[left-1];
    }
};

/**
 * Your NumArray object will be instantiated and called as such:
 * NumArray* obj = new NumArray(nums);
 * int param_1 = obj->sumRange(left,right);
 */
```

QUESTION :2

2. Given an array of integers `nums`, calculate the **pivot index** of this array.

   The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

   If the index is on the left edge of the array, then the left sum is `0` because there are no elements to the left. This also applies to the right edge of the array.

   Return *the **leftmost pivot index***. If no such index exists, return `-1`.
   [Leetcode 724]

Answer:

```
class Solution {
public:
 int pivotIndex(vector<int>& a) {
 int n = a.size();
 int leftsum = 0 , rightsum = 0;
 for(auto x:a)rightsum += x;
 for(int i=0;i<n;i++){
 rightsum = rightsum - a[i];
 if(leftsum == rightsum)return i;
 leftsum += a[i];
 }
 return -1;
 }
};
```

Question:3

3. We define the **conversion array** `conver` of an array `arr` as follows:

Answer:

```
class Solution {
public:
 vector<long long> findPrefixScore(vector<int>& a) {
 int n = a.size();
 vector<long long int>res(n,0);
 res[0] = 2*a[0];
 int maxi = a[0];
 // maxi = max(maxi , a[0]);
 for(int i=1;i<n;i++){
 maxi = max(maxi , a[i]);
 res[i] = a[i] + maxi + res[i-1];
 }
 return res;
 }
};
```

Question:4

4. There are `n` flights that are labeled from `1` to `n`.

   You are given an array of flight bookings `bookings`, where `bookings[i] = [firsti, lasti, seatsi]` represents a booking for flights `firsti` through `lasti` (**inclusive**) with `seatsi` seats reserved for **each flight** in the range.

Answer:

```
class Solution {
public:
 vector<int> corpFlightBookings(vector<vector<int>>& a, int n) {
 vector<int>res(n,0);
 for(int i=0;i<a.size();i++){
 res[a[i][0] - 1] += a[i][2];
 if(a[i][1] < n)res[a[i][1]] -= a[i][2];
 }
 for(int i=1;i<n;i++){
 res[i] += res[i-1];
 }
 return res;
```

```
  }
};
```