LeetCode: 1480

✅ Problem: **1480. Running Sum of 1d Array**

👉 Given an array `nums`, mujhe ek **running sum array** return karna hai jahan:

```
runningSum[i] = sum(nums[0] + nums[1] + ... + nums[i])
```

🧠 Mera Approach

- Mujhe har index tak ka **sum calculate** karna hai aur ek naye vector me store karna hai.
- Sochne wali baat:
  - ✅ Direct sum nikalne ke liye mujhe har step par pichle elements ko baar-baar add nahi karna ($O(n^2)$ nahi chahiye).
  - ✅ Instead, main ek **running total (sum)** rakhta hoon aur har step par update karta hoon.
- **Plan**:
  1. Ek variable `sum` banata hoon, initially 0.
  2. Loop chalayunga array ke through.
  3. Har element ko `sum` me add karunga.
  4. Is updated `sum` ko `ans` vector me dal dunga.
  5. Finally `ans` return kar dunga.

✅ Mera Code

```
class Solution {
public:
    vector<int> runningSum(vector<int>& nums) {
        int sum = 0;              // 🟢 Step 1: running total start with 0
        int n = nums.size();      // 🟢 Step 2: length of nums
        vector<int> ans(n, 0);    // 🟢 Step 3: result array of size n initialized wit
h 0

        for(int i = 0; i < n; i++){
            sum += nums[i];       // 🟢 Step 4: add current element to running su
m
            ans[i] = sum;         // 🟢 Step 5: store current sum at index i
        }
        return ans;               // 🟢 Step 6: return the result
    }
};
```

🔍 Dry Run Example

Input:

```
nums = [1, 2, 3, 4]
```

Steps:

- sum = 0, ans = [0, 0, 0, 0]
- i=0 → sum = 0 + 1 = 1 → ans[0] = 1 ✅
- i=1 → sum = 1 + 2 = 3 → ans[1] = 3 ✅
- i=2 → sum = 3 + 3 = 6 → ans[2] = 6 ✅
- i=3 → sum = 6 + 4 = 10 → ans[3] = 10 ✅

Output:

```
[1, 3, 6, 10]
```

🔑 Key Points

✔️ Time Complexity = **O(n)** (sirf ek loop)

✔️ Space Complexity = **O(n)** (result store karne ke liye)

✔️ In-place bhi kar sakte the agar nums me hi update karte.

💡 Alternative Idea

- Agar constraint allow kare, hum **nums me hi changes kar sakte hain**:

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

Space: O(1) ✅
