Subarray with 0 sum in C++

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

int ZeroSumSubarray(vector<int>& arr) {
    unordered_set<int> us;
    int prefix_sum = 0;
    us.insert(0); // Insert 0 initially to handle cases where the prefix_sum itself is zero
    for (int i = 0; i < arr.size(); ++i) {
        prefix_sum += arr[i];
        if (us.count(prefix_sum) > 0)
            return 1; // Found a subarray with sum zero
        us.insert(prefix_sum);
    }
    return 0; // No subarray with sum zero found
}

int main() {
    vector<int> arr = {5, 3, 9, -4, -6, 7, -1};
    cout << ZeroSumSubarray(arr) << endl;
    return 0;
}
```

Dry Run of `ZeroSumSubarray(arr)`

Input:

```
arr = {5, 3, 9, -4, -6, 7, -1};
```

Step 1: Initialize Variables

- **Prefix Sum (`prefix_sum`)** = `0`
- **Hash Set (`us`)** = `{0}` (We insert 0 initially to handle cases where the prefix sum itself is zero)

Step 2: Iterating Over the Array

Iteration	`arr[i]`	`prefix_sum` (cumulative)	`us` (hash set)	Check if `prefix_sum` exists in `us`
1	5	0 + 5 = 5	{0, 5}	No
2	3	5 + 3 = 8	{0, 5, 8}	No
3	9	8 + 9 = 17	{0, 5, 8, 17}	No
4	-4	17 - 4 = 13	{0, 5, 8, 17, 13}	No
5	-6	13 - 6 = 7	{0, 5, 8, 17, 13, 7}	No
6	7	7 + 7 = 14	{0, 5, 8, 17, 13, 7, 14}	No
7	-1	14 - 1 = 13	{0, 5, 8, 17, 13, 7, 14}	**Yes (13 exists in set!)**

Step 3: Return Result

- Since `prefix_sum = 13` **already exists in**

	us, it means there exists a subarray with sum `0`. • **Return `1` (True)**.
Output: 1	