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;
}
```

Input:

vector<int> arr = {5, 3, 9, -4, -6, 7, -1};

Goal:

Check whether there exists a subarray whose sum is zero.

Key Concepts:

- **Prefix Sum**: It is the cumulative sum of elements up to the current index.
- **Hash Set (unordered_set)**: Used to store the prefix sums encountered so far. If a prefix sum repeats, it means the sum of elements between these two indices is zero.

Step-by-Step Execution:

1. **Initialization:**
    - We initialize an unordered set us to store the prefix sums, starting by inserting 0 into it (this helps in case the sum of elements from the start up to the current element is zero).
    - prefix_sum is initialized to 0.
2. **Loop through the array:**
    - We iterate over the array, computing the prefix sum at each step.

Iteration 1:

- i = 0: arr[i] = 5
- prefix_sum = 0 + 5 = 5
- Check if prefix_sum = 5 exists in the set. It doesn't, so we insert 5 into the set.

Set us: {0, 5}

Iteration 2:

- i = 1: arr[i] = 3
- prefix_sum = 5 + 3 = 8
- Check if prefix_sum = 8 exists in the set. It doesn't, so we insert 8 into the set.

Set us: {0, 5, 8}

Iteration 3:

- i = 2: arr[i] = 9
- prefix_sum = 8 + 9 = 17
- Check if prefix_sum = 17 exists in the set. It doesn't, so we insert 17 into the set.

Set us: {0, 5, 8, 17}

Iteration 4:

- i = 3: arr[i] = -4
- prefix_sum = 17 + (-4) = 13
- Check if prefix_sum = 13 exists in the set. It doesn't, so we insert 13 into the set.

Set us: {0, 5, 8, 13, 17}

Iteration 5:

- i = 4: arr[i] = -6
- prefix_sum = 13 + (-6) = 7
- Check if prefix_sum = 7 exists in the set. It doesn't, so we insert 7 into the set.

Set us: {0, 5, 7, 8, 13, 17}

Iteration 6:

- i = 5: arr[i] = 7
- prefix_sum = 7 + 7 = 14
- Check if prefix_sum = 14 exists in the set. It doesn't, so we insert 14 into the set.

Set us: {0, 5, 7, 8, 13, 14, 17}

Iteration 7:

- i = 6: arr[i] = -1
- prefix_sum = 14 + (-1) = 13
- Check if prefix_sum = 13 exists in the set. It **does** exist (it was added in iteration 4).

Since prefix_sum = 13 is found in the set, it means there is a subarray between index 4 and index 6 whose sum is zero. Therefore, we return 1.

	Final Output: 1
Output: 1	