Subarray with given sum in C++

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

bool isSum(int arr[], int n, int sum) {
    unordered_set<int> s;
    int pre_sum = 0;
    for (int i = 0; i < n; i++) {
        if (pre_sum == sum) {
            return true;
        }
        pre_sum += arr[i];
        if (s.find(pre_sum - sum) != s.end()) {
            return true;
        }
        s.insert(pre_sum);
    }
    return false;
}

int main() {
    int arr[] = {5, 8, 6, 13, 3, -1};
    int sum = 22;
    int n = sizeof(arr) / sizeof(arr[0]);

    if (isSum(arr, n, sum)) {
        cout << "Subarray with sum " << sum << "
exists." << endl;
    } else {
        cout << "No subarray with sum " << sum << "
exists." << endl;
    }

    return 0;
}
```

Input:

```
int arr[] = {5, 8, 6, 13, 3, -1};
int sum = 22;
```

Goal:

Check whether there exists a subarray whose sum equals 22.

Logic Explanation:

The isSum function tries to find a subarray with a sum equal to sum (in this case, 22). It uses a **prefix sum** approach combined with a **hash set** to track the cumulative sums encountered so far.

1. pre_sum: Keeps track of the cumulative sum of elements as we iterate through the array.
2. We check if the difference between the pre_sum and the sum (i.e., pre_sum - sum) has already been encountered. If it has, then there exists a subarray with the required sum.
3. We insert each cumulative sum into a set (s) to help with the lookup.

Step-by-Step Execution:

1. **Initialization:**
    - We initialize pre_sum = 0 and an empty unordered set s.
2. **Iteration 1 (i = 0):**
    - arr[i] = 5
    - pre_sum = 0 + 5 = 5
    - Check if pre_sum == sum. It's not (5 != 22).
    - Check if pre_sum - sum = 5 - 22 = -17 is in the set. It is not.
    - Insert pre_sum = 5 into the set.
    - Set s = {5}
3. **Iteration 2 (i = 1):**
    - arr[i] = 8
    - pre_sum = 5 + 8 = 13
    - Check if pre_sum == sum. It's not (13 != 22).
    - Check if pre_sum - sum = 13 - 22 = -9 is in the set. It is not.
    - Insert pre_sum = 13 into the set.
    - Set s = {5, 13}
4. **Iteration 3 (i = 2):**

	○ arr[i] = 6 ○ pre_sum = 13 + 6 = 19 ○ Check if pre_sum == sum. It's not (19 != 22). ○ Check if pre_sum - sum = 19 - 22 = -3 is in the set. It is not. ○ Insert pre_sum = 19 into the set. ○ Set s = {5, 13, 19} 5. **Iteration 4 (i = 3):** ○ arr[i] = 13 ○ pre_sum = 19 + 13 = 32 ○ Check if pre_sum == sum. It's not (32 != 22). ○ Check if pre_sum - sum = 32 - 22 = 10 is in the set. It is not. ○ Insert pre_sum = 32 into the set. ○ Set s = {5, 13, 19, 32} 6. **Iteration 5 (i = 4):** ○ arr[i] = 3 ○ pre_sum = 32 + 3 = 35 ○ Check if pre_sum == sum. It's not (35 != 22). ○ Check if pre_sum - sum = 35 - 22 = 13 is in the set. **It is!** ○ Since 13 exists in the set, it means that the sum of the subarray from index 2 to index 4 equals 22. We return true. **Conclusion:** The code returns true because a subarray with sum 22 exists, specifically the subarray {6, 13, 3}. **Final Output:** Subarray with sum 22 exists.
Output: Subarray with sum 22 exists.	