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;
}
```

Dry Run of `isSum()` Function

Input:

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

Step 1: Initialize Variables

- **Prefix Sum (`pre_sum`)** = `0`
- **Hash Set (`s`)** = `{}` (Empty initially)

Step 2: Iterating Over the Array

Iteration	`arr[i]`	`pre_sum` (cumulative)	`pre_sum - sum`	Check if `pre_sum - sum` exists in set	Update Hash Set
1	5	0 + 5 = 5	5 - 22 = -17	No	{5}
2	8	5 + 8 = 13	13 - 22 = -9	No	{5, 13}
3	6	13 + 6 = 19	19 - 22 = -3	No	{5, 13, 19}
4	13	19 + 13 = 32	32 - 22 = 10	No	{5, 13, 19, 32}
5	3	32 + 3 = 35	35 - 22 = 13	**Yes** (13 exists in set)	{5, 13, 19, 32, 35}
6	-1	35 + (-1) = 34	34 - 22 = 12	No	{5, 13, 19, 32, 35, 34}

Step 3: Return Result

- At iteration 5, when `pre_sum = 35`, `pre_sum - sum = 13` is found in the hash set, which means there exists a subarray with a sum of `22`.
- **Return `true`**.

Output: Subarray with sum 22 exists.	