2.Medium\05.Number_of_subarray_sum_equal_k.cpp

```cpp
/*
QUESTION:
Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k.

Example:
Input: nums = [1,1,1], k = 2
Output: 2

APPROACH:
To find the total number of subarrays with sum equal to k, we can use the technique of prefix sum along with a hashmap.
1. Initialize a variable `count` to keep track of the count of subarrays with sum equal to k.
2. Initialize a variable `prefixSum` to keep track of the prefix sum while iterating through the array.
3. Initialize a hashmap `sumCount` to store the frequency of prefix sums encountered so far.
4. Set the initial prefix sum to 0 and set its count to 1 in the `sumCount` hashmap.
5. Iterate through the array and update the prefix sum by adding each element.
6. Check if the current prefix sum minus k exists in the `sumCount` hashmap. If it does, add the count of that prefix sum to the `count` variable.
7. Increment the count of the current prefix sum in the `sumCount` hashmap.
8. Finally, return the `count` variable as the total number of subarrays with sum equal to k.

CODE:
*/

int subarraySum(vector<int> &nums, int k)
{
    int pref_sum = 0;
    unordered_map<int, int> mp;
    int ans = 0;

    for (int i = 0; i < nums.size(); i++)
    {
        pref_sum += nums[i];

        if (pref_sum == k)
            ans++;

        if (mp.find(pref_sum - k) != mp.end())
        {
            ans += mp[pref_sum - k];
        }

        mp[pref_sum]++;
    }

    return ans;
}

/*
TIME COMPLEXITY: O(n), where n is the size of the input array nums.
```

```
SPACE COMPLEXITY: O(n), as we are using a hashmap to store the prefix sums and their corresponding counts.
*/

```