3.Hard\13.Longest_subarray_with_sum_k_containg_+ves_and_-ves.cpp

```cpp
/*Question:
Given an array containing N integers and an integer K, find the length of the longest subarray with the sum of the elements equal to K.

Example:
Input:
A[] = {10, 5, 2, 7, 1, 9}
K = 15
Output:
4
Explanation:
The sub-array is {5, 2, 7, 1}.

Approach:
To solve this problem, we can use a prefix sum approach along with a hashmap to keep track of the prefix sums encountered so far. We iterate through the array and maintain a prefix sum variable. At each index, we check if the prefix sum equals K, in which case we update the maximum length of the subarray found so far. Additionally, we check if the current prefix sum minus K exists in the hashmap. If it does, it means there is a subarray between the previous occurrence of the prefix sum minus K and the current index that sums up to K. We update the maximum length accordingly. We store the prefix sums and their corresponding indices in the hashmap.

Code:
*/
int lenOfLongSubarr(int A[], int N, int K) {
    int pref_sum = 0;
    int ans = 0;
    unordered_map<int, int> mp;

    for (int i = 0; i < N; i++) {
        pref_sum += A[i];
        if (pref_sum == K)
            ans = max(ans, i + 1);
        if (mp.find(pref_sum - K) != mp.end())
            ans = max(ans, i - mp[pref_sum - K]);
        if (mp.find(pref_sum) == mp.end())
            mp[pref_sum] = i;
    }
    return ans;
}
/*
Time Complexity: The code iterates through the array once, resulting in a time complexity of O(N), where N is the size of the array.
Space Complexity: The code uses an unordered map to store the prefix sums and their corresponding indices. In the worst case, all elements of the array could be distinct, leading to a space complexity of O(N) to store the prefix sums in the map.
*/
```