Longest Substring With At Most K Unique Characters in C++

```cpp
#include <iostream>
#include <string>
#include <unordered_map>

class LongestSubstringWithAtMostKUniqueCharacters {
public:
    static int sol(const std::string& str, int k) {
        int ans = 0;
        int i = -1;
        int j = -1;
        std::unordered_map<char, int> map;

        while (true) {
            bool f1 = false;
            bool f2 = false;

            while (i < static_cast<int>(str.length()) - 1) {
                f1 = true;
                i++;
                char ch = str[i];
                map[ch]++;

                if (map.size() <= k) {
                    int len = i - j;
                    if (len > ans) {
                        ans = len;
                    }
                } else {
                    break;
                }
            }

            while (j < i) {
                f2 = true;
                j++;
                char ch = str[j];
                if (map[ch] == 1) {
                    map.erase(ch);
                } else {
                    map[ch]--;
                }

                if (map.size() > k) {
                    continue;
                } else {
                    int len = i - j;
                    if (len > ans) {
                        ans = len;
                    }
                    break;
                }
            }

            if (!f1 && !f2) {
                break;
            }
        }
        return ans;
    }
};
```

Understanding the Problem

- The function **`sol(str, k)`** finds the **longest substring** with at most `k` unique characters.
- Uses **two-pointer sliding window** technique (`i` and `j`) with an **unordered_map** to track character frequencies.
- Expands the window until the number of unique characters exceeds `k`, then shrinks the window.

Example Input

```
string str = "ddacbbaccdedacebb";
int k = 3;
```

Expected Output: `7`

Step-by-Step Dry Run

Step	i	j	Window (`str[j+1]` to `str[i]`)	Unique Chars	Max Length (ans)
1	0	-1	`d`	1	1
2	1	-1	`dd`	1	2
3	2	-1	`dda`	2	3
4	3	-1	`ddac`	3	4
5	4	-1	`ddacb`	4 (exceeds k)	4
6	4	0	`dacb`	3	4
7	5	0	`dacbb`	3	5
8	6	0	`dacbba`	3	6
9	7	0	`dacbbac`	3	7 ✓
10	8	0	`dacbbacc`	3	7
11	9	1	`acbbaccd`	4 (exceeds k)	7
12	9	2	`cbbaccd`	3	7
13	10	2	`cbbaccde`	4 (exceeds k)	7
14	10	3	`bbaccde`	3	7
15	11	3	`bbaccded`	4 (exceeds	7

```
int main() {
    std::string str = "ddacbbaccdedacebb";
    int k = 3;
    std::cout <<
LongestSubstringWithAtMostKUniqueCharacters::sol(str
, k) << std::endl;
    return 0;
}
```

				k)	
...	...	...	...	...	...

Final Output

✅ **Longest substring with at most `k = 3` unique characters: 7**

Output:-
7