Count Palindromic Subsequence C++

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

int
countPalindromicSubseq(const
string& str) {
    int n = str.length();
    int dp[n][n] = {0};  //
Initialize the 2D array

    for (int g = 0; g < n; g++) {
        for (int i = 0, j = g; j < n;
i++, j++) {
            if (g == 0) {
                dp[i][j] = 1;
            } else if (g == 1) {
                dp[i][j] = (str[i] ==
str[j]) ? 2 : 1;
            } else {
                if (str[i] == str[j]) {
                    dp[i][j] = dp[i][j -
1] + dp[i + 1][j] + 1;
                } else {
                    dp[i][j] = dp[i][j -
1] + dp[i + 1][j] - dp[i + 1][j -
1];
                }
            }
        }
    }

    return dp[0][n - 1];
}

int main() {
    string str = "abccbc";
    cout <<
countPalindromicSubseq(str)
<< endl;
    return 0;
}
```

Step 1: Single Character (g = 0)

Each **single character** is a palindrome:

dp[i][i] = 1

Updated DP Table:

1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1

Step 2: Two-Character Substrings (g = 1)

i	j	Substring	str[i] == str[j]?	dp[i][j]
0	1	"ab"	✗	1
1	2	"bc"	✗	1
2	3	"cc"	✓	2
3	4	"cb"	✗	1
4	5	"bc"	✗	1

Updated DP Table:

1 1 0 0 0 0
0 1 1 0 0 0
0 0 1 2 0 0
0 0 0 1 1 0
0 0 0 0 1 1
0 0 0 0 0 1

Step 3: Three-Character Substrings (g = 2)

i	j	Substring	str[i] == str[j]?	Formula Used	dp[i][j]
0	2	"abc"	✗	dp[0][2] = dp[0][1] + dp[1][2] - dp[1][1]	2
1	3	"bcc"	✗	dp[1][3] = dp[1][2] + dp[2][3] - dp[2][2]	3
2	4	"ccb"	✗	dp[2][4] = dp[2][3] + dp[3][4] - dp[3][3]	3
3	5	"cbc"	✓	dp[3][5] = dp[3][4] + dp[4][5] + 1	3

Updated DP Table:

1 1 2 0 0 0
0 1 1 3 0 0
0 0 1 2 3 0
0 0 0 1 1 3
0 0 0 0 1 1

0 0 0 0 0 1

Step 4: Four-Character Substrings (g = 3)

i	j	Substring	str[i] == str[j]?	Formula Used	dp[i][j]
0	3	"abcc"	✗	dp[0][3] = dp[0][2] + dp[1][3] - dp[1][2]	4
1	4	"bccb"	✓	dp[1][4] = dp[1][3] + dp[2][4] + 1	7
2	5	"ccbc"	✓	dp[2][5] = dp[2][4] + dp[3][5] + 1	7

Updated DP Table:

1 1 2 4 0 0
0 1 1 3 7 0
0 0 1 2 3 7
0 0 0 1 1 3
0 0 0 0 1 1
0 0 0 0 0 1

Step 4: Four-Character Substrings (g = 4)

i	j	Substring	str[i] == str[j]?	Formula Used	dp[i][j]
0	4	"abccb"	✗	dp[0][4] = dp[0][3] + dp[1][4] - dp[1][3]	5
1	5	"bccbc"	✓	dp[1][5] = dp[1][4] + dp[2][5] + 1	9

Updated DP Table:

1 1 2 4 5 0
0 1 1 3 7 9
0 0 1 2 3 7
0 0 0 1 1 3
0 0 0 0 1 1
0 0 0 0 0 1

Step 5: Final Computation (g = 5)

dp[0][5]=dp[0][4]+dp[1][5]−dp[1][4]dp[0][5] = dp[0][4] + dp[1][5] - dp[1][4]dp[0][5]=dp[0][4]+dp[1][5]−dp[1][4] dp[0][5]=7+7−5=9dp[0][5] = 7 + 7 - 5 = 9dp[0][5]=7+7−5=9

Output:-
9