Count of strings where adjacent characters are of difference one

iven a number n, count number of strings of length n such that every string has adjacent characters with difference between ASCII values as 1.

**Examples**:

Input : N = 1

Output : Total strings are 26

Explanation : For N=1, strings

are a, b, c,, ...., x, y, z

Input : N = 2

Output : Total strings are 50

Explanation : For N = 2, strings

are ab, ba, bc, cb, .., yx, yz, zy

[**Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.**](https://ide.geeksforgeeks.org/)

For strings starting with character ‘A’ and length ‘i’, we consider all strings of length ‘i-1’ and starting with character ‘B’

For strings starting with character ‘G’ and length ‘i’, we consider all strings of length ‘i-1’ and starting with character ‘H’ and all strings of length ‘i-1’ and starting with ‘F’.

We take the base case for n = 1, and set result for all 26 characters as 1. This simply means when 1 character string is consider all alphabets from a-z are taken only once.

For **N = 2**,

For **N = 3**,

**Conclusion** : For **N = n**

**countAdjacent(n)**

dp[i][j] finally stores count of strings

of length i and starting with

character j.

Initialize dp[n+1][27] as 0

Initialize dp[1][j] = 1 where j = 0 to 25

for i = 2 to n

for j = 0 to 25

if (j = 0)

dp[i][j] = dp[i-1][j+1];

else

dp[i][j] = dp[i-1][j-1] + dp[i-1][j+1];

Sum of n-th row from 0 to 25 is the result.

Solution:

// CPP Program to count strings with adjacent

// characters.

#include <bits/stdc++.h>

using namespace std;

int countStrs(int n)

{

    long int dp[n + 1][27];

    // Initializing arr[n+1][27] to 0

    memset(dp, 0, sizeof(dp));

    // Initialing 1st row all 1 from 0 to 25

    for (int i = 0; i <= 25; i++)

        dp[1][i] = 1;

    // Begin evaluating from i=2 since 1st row is set

    for (int i = 2; i <= n; i++) {

        for (int j = 0; j <= 25; j++)

            // j=0 is 'A' which can make strings

            // of length i using strings of length

            // i-1 and starting with 'B'

            if (j == 0)

                dp[i][j] = dp[i - 1][j + 1];

            else

                dp[i][j] = (dp[i - 1][j - 1] +

                            dp[i - 1][j + 1]);

    }

    // Our result is sum of last row.

    long int sum = 0;

    for (int i = 0; i <= 25; i++)

        sum = (sum + dp[n][i]);

    return sum;

}

// Driver's Code

int main()

{

    int n = 3;

    cout << "Total strings are : " << countStrs(n);

    return 0;

}