**[DYNAMIC PROGRAMMING](http://118.185.187.137/moodle/course/view.php?id=158" \l "section-1)**

**AIM:**

To implement an efficient algorithm that calculates the number of ways a positive integer n can be represented using the numbers 1 and 3.

**PROGRAM:**

#include<stdio.h>

long long count(int n)

{

long long d[n+1];

for(int i=0;i<=n;i++)

{

d[i]=0;

}

d[0]=1;

if(n>=1){d[1]=1;}

if(n>=2){d[2]=1;}

if(n>=3){d[3]=2;}

for(int i=4;i<=n;i++)

{

d[i]=d[i-1]+(i>=3?d[i-3]:0);

}

return d[n];

}

int main()

{

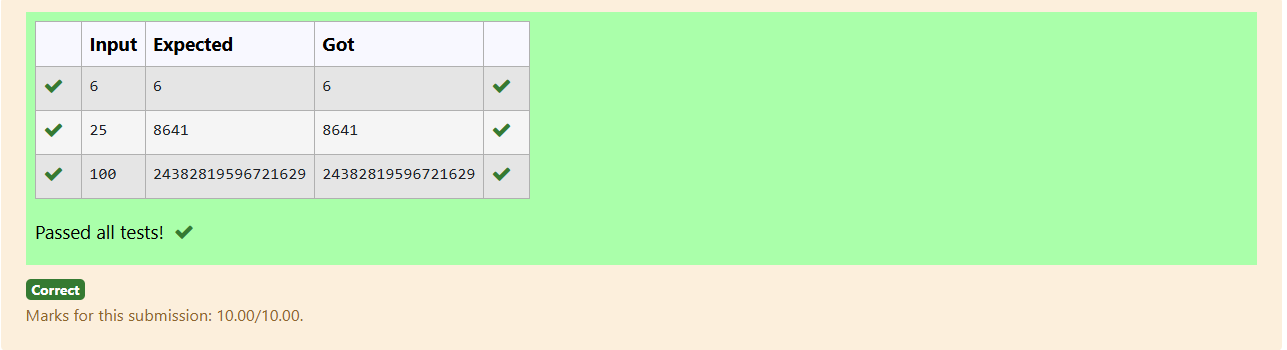
int n;

scanf("%d",&n);

printf("%lld",count(n));

}

**OUTPUT:**



**AIM:**

To implement a dynamic programming algorithm that helps Ram find the maximum monetary path from the top-left to the bottom-right corner on an n x n chessboard, moving only right or down.

**PROGRAM:**

#include <stdio.h>

#define MAX\_N 100

int main() {

int n;

scanf("%d", &n);

int arr[MAX\_N][MAX\_N];

int dp[MAX\_N][MAX\_N];

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

scanf("%d", &arr[i][j]);

}

}

dp[0][0] = arr[0][0]

for (int j = 1; j < n; j++) {

dp[0][j] = dp[0][j - 1] + arr[0][j];

}

for (int i = 1; i < n; i++) {

dp[i][0] = dp[i - 1][0] + arr[i][0];

}

for (int i = 1; i < n; i++) {

for (int j = 1; j < n; j++) {

dp[i][j] = (dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1]) + arr[i][j];

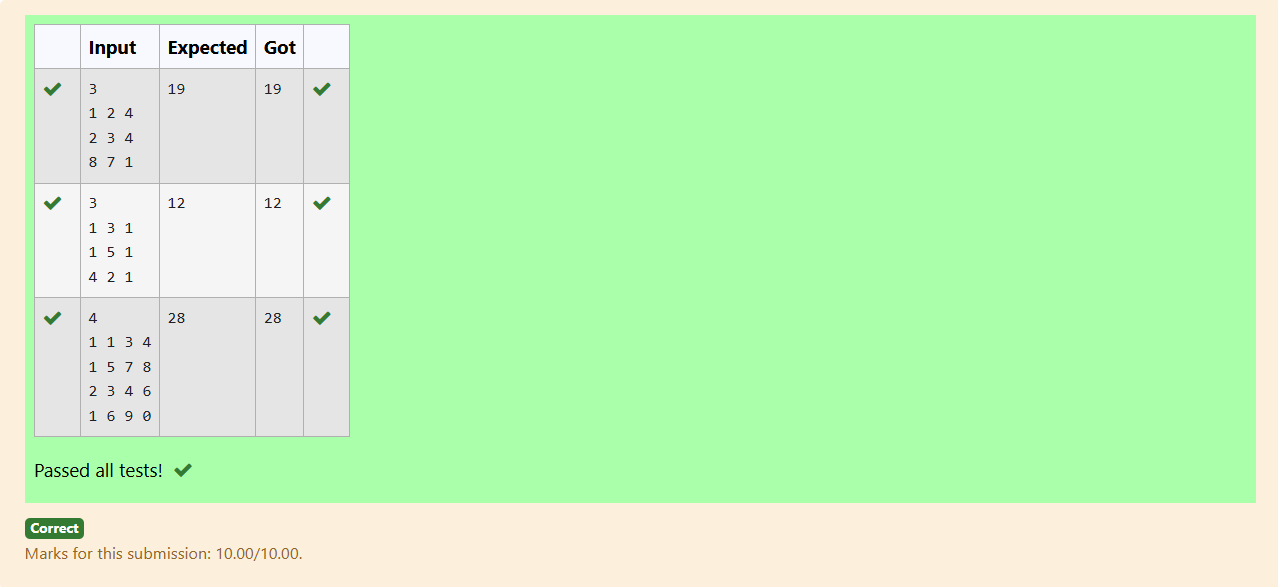
}

}

printf("%d\n", dp[n - 1][n - 1]);

return 0; }

**OUTPUT:**



**AIM:**

To implement an algorithm that finds the length of the longest common subsequence between two given strings.

**PROGRAM:**

#include <stdio.h>

#include <string.h>

#define MAX\_LEN 1000

int longestCommonSubsequence(char \*s1, char \*s2) {

int m = strlen(s1);

int n = strlen(s2);

int dp[MAX\_LEN + 1][MAX\_LEN + 1];

for (int i = 0; i <= m; i++) {

for (int j = 0; j <= n; j++) {

if (i == 0 || j == 0) {

dp[i][j] = 0;

}

else if (s1[i - 1] == s2[j - 1]) {

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

}

else {

dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];

}

}

}

return dp[m][n];

}

int main() {

char s1[MAX\_LEN], s2[MAX\_LEN];

scanf("%s", s1);

scanf("%s", s2);

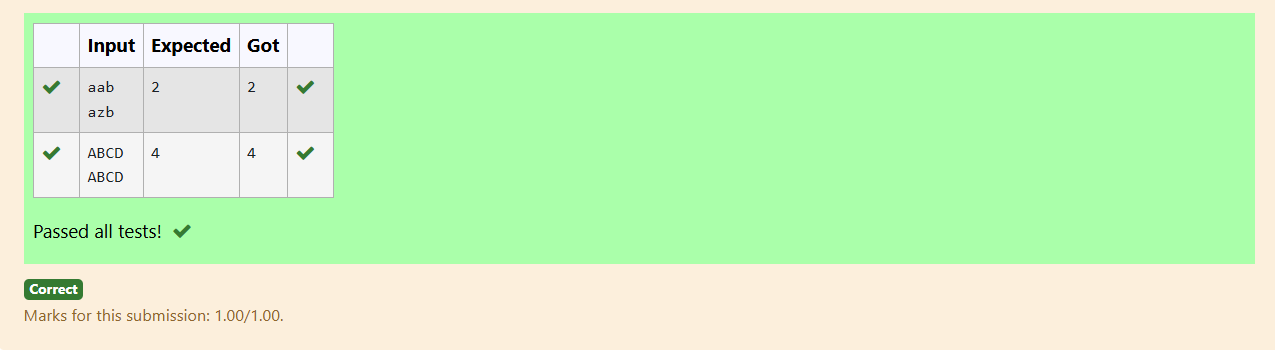
int lcsLength = longestCommonSubsequence(s1, s2);

printf("%d",lcsLength);

return 0;

}

**OUTPUT:**



**AIM:**

To implement an algorithm that determines the length of the longest non-decreasing subsequence in a given sequence.

**PROGRAM:**

#include <stdio.h>

int longestNonDecreasingSubsequence(int sequence[], int n) {

int dp[n];

int maxLength = 1;

for (int i = 0; i < n; i++) {

dp[i] = 1;

}

for (int i = 1; i < n; i++) {

for (int j = 0; j < i; j++) {

if (sequence[j] <= sequence[i]) {

dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : (dp[j] + 1);

}

}

maxLength = (maxLength > dp[i]) ? maxLength : dp[i];

}

return maxLength;

}

int main() {

int n;

scanf("%d", &n);

int sequence[n];

for (int i = 0; i < n; i++) {

scanf("%d", &sequence[i]);

}

int length = longestNonDecreasingSubsequence(sequence, n);

printf("%d",length);

return 0; }

**OUTPUT:**

