DYNAMIC PROGRAMMING PROBLEMS

**Playing with Numbers:**

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer ‘n’ and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

**Example 1:**

***Input:****6****Output:****6****Explanation:****There are 6 ways to 6 represent number with 1 and 3  
         1+1+1+1+1+1  
         3+3  
         1+1+1+3  
         1+1+3+1  
         1+3+1+1  
         3+1+1+1*  
**Input Format**  
First Line contains the number n  
   
**Output Format**

**Print: The number of possible ways ‘n’ can be represented using 1 and 3**

Sample Input  
   
6

Sample Output

6

PROGRAM:

#include <stdio.h>

long long int dp(int a){

long long int arr[a+1];

arr[0]=0;

arr[1]=1;

if(a>=2)arr[2]=1;

if(a>=3)arr[3]=2;

for (int i=4;i<=a;i++){

arr[i]=arr[i-1]+arr[i-3];

}

return arr[a];

}

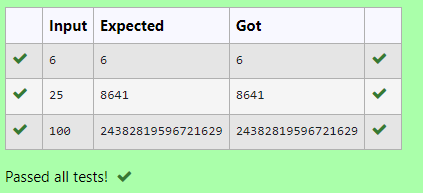
int main (){

int a;

scanf("%d",&a);

printf("%lld",dp(a));

}



2. **Playing with Chessboard:**

Ram is given with an n\*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

**Example:**  
**Input**  
3  
**1** 2 4  
**2**3 4  
**8 7 1**  
**Output:**  
19

**Explanation:**  
Totally there will be 6 paths among that the optimal is  
 Optimal path value:1+2+8+7+1=19

**Input Format**  
First Line contains the integer n  
The next n lines contain the n\*n chessboard values  
   
**Output Format**

Print Maximum monetary value of the path

PROGRAM:

#include <stdio.h>

int dpfunc(int arr[100][100],int n){

int dp[100][100];

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

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

int main(){

int n;scanf("%d", &n);

int board[100][100];

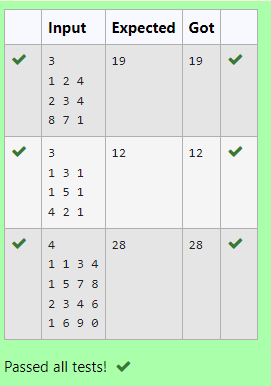
for(int i=0;i<n;i++){

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

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

printf("%d\n",dpfunc(board,n));

return 0;}



3. Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

 s1: ggtabe

 s2: tgatasb

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s1 |  | a | g | **g** | **t** | **a** | **b** |  |
| s2 |  | **g** | x | **t** | x | **a** | y | **b** |

**The length is 4**

Solveing it using Dynamic Programming

**For example:**

| **Input** | **Result** |
| --- | --- |
| aab  azb | 2 |

PROGRAM:

#include <stdio.h>

#include <string.h>

#define MAX 100

typedef struct{

char data[MAX];} String;

int max(int a,int b){

return(a>b)?a:b;}

int lcs(String X, String Y){

int m = strlen(X.data);

int n = strlen(Y.data);

int L[MAX][MAX]={0};

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

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

if(i==0||j==0)L[i][j] = 0;

else if(X.data[i-1]==Y.data[j-1])L[i][j]=L[i-1][j-1]+1;

else L[i][j]=max(L[i-1][j],L[i][j-1]);}}

return L[m][n];}

int main() {

String X, Y;

fgets(X.data, MAX, stdin);

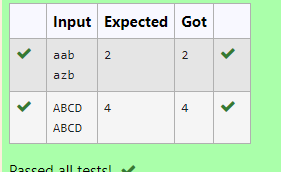
X.data[strcspn(X.data,"\n")]='\0';

fgets(Y.data, MAX, stdin);

Y.data[strcspn(Y.data,"\n")]='\0';

printf("%d\n", lcs(X, Y));

return 0;}



4. Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

PROGRAM:

#include <stdio.h>

int max(int a,int b){

return (a > b) ? a : b;}

int lnds(int arr[],int n){

int lnds[n],m=1;

for(int i=0;i<n;i++)lnds[i]=1;

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

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

if(arr[i]>=arr[j])lnds[i]=max(lnds[i],lnds[j]+1);}

m=max(m,lnds[i]);}

return m;}

int main() {

int n;scanf("%d", &n);

int arr[n];

for(int i=0;i<n;i++)scanf("%d",&arr[i]);

printf("%d\n",lnds(arr,n));

return 0;}

