Question **1**

Correct

Mark 10.00 out of 10.00

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int main() {
    long n;
    scanf("%ld", &n);
    long dp[n + 1];
    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    dp[3] = 2;
    for (int i = 4; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 3];
    }
    printf("%ld",dp[n]);
    return 0;
}

```

	Input	Expected	Got	
✓	6	6	6	✓
✓	25	8641	8641	✓
✓	100	24382819596721629	24382819596721629	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

◄ 5-G-Product of Array elements-Minimum

Jump to...

2-DP-Playing with chessboard ►

Question **1**

Correct

Mark 10.00 out of 10.00

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

Answer: (penalty regime: 0 %)

```
//     }
//     if((arr[0][0]+arr[0][1]+arr[1][1]+arr[2][1]+arr[2][2])>s){
//         s=arr[0][0]+arr[0][1]+arr[1][1]+arr[2][1]+arr[2][2];
//         printf("sad");
//     }
//     if((arr[0][0]+arr[1][0]+arr[1][1]+arr[2][1]+arr[2][2])>s){
//         s=arr[0][0]+arr[1][0]+arr[1][1]+arr[2][1]+arr[2][2];
//         printf("sad");
//     }
//     if((arr[0][0]+arr[0][1]+arr[0][2]+arr[1][2]+arr[1][1]+arr[1][0]+arr[2][0]+arr[2][1]+arr[2][2])>s){
//         s=arr[0][0]+arr[0][1]+arr[0][2]+arr[1][2]+arr[1][1]+arr[1][0]+arr[2][0]+arr[2][1]+arr[2][2];
//         printf("sad");
//     }
//     if((arr[0][0]+arr[1][0]+arr[1][1]+arr[1][2]+arr[2][2])>s){
//         s=arr[0][0]+arr[1][0]+arr[1][1]+arr[1][2]+arr[2][2];
//         printf("sadlad");
//     }
//     printf("%d",s);

 //}
 #include <stdio.h>

#define MAX_SIZE 100

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

    int grid[MAX_SIZE][MAX_SIZE];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &grid[i][j]);
        }
    }
    int dp[MAX_SIZE][MAX_SIZE];
    dp[0][0] = grid[0][0];
    for (int j = 1; j < n; j++) {
        dp[0][j] = dp[0][j - 1] + grid[0][j];
    }
    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i - 1][0] + grid[i][0];
    }
    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            dp[i][j] = grid[i][j] + (dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1]);
        }
    }
    printf("%d\n", dp[n - 1][n - 1]);

    return 0;
}

```

	Input	Expected	Got	
✓	3 1 2 4 2 3 4 8 7 1	19	19	✓
✓	3 1 3 1 1 5 1 4 2 1	12	12	✓
✓	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

◄ 1-DP-Playing with Numbers

Jump to...

3-DP-Longest Common Subsequence ►

Question **1**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <string.h>
int main() {
    char s1[100], s2[100];
    scanf("%s", s1);
    scanf("%s", s2);
    int l1 = strlen(s1);
    int l2 = strlen(s2);
    int arr[l1+1][l2+1];
    for (int i=0;i<=l1;i++) {
        for (int j=0;j<=l2;j++) {
            if (i==0 || j==0) {
                arr[i][j]=0;
            } else if (s1[i-1]==s2[j-1]) {
                arr[i][j]=arr[i-1][j-1]+1;
            } else {
                arr[i][j]=(arr[i-1][j]>arr[i][j-1])?arr[i-1][j]:arr[i][j-1];
            }
        }
    }
    printf("%d",arr[l1][l2]);
    return 0;
}
```

	Input	Expected	Got	
✓	aab azb	2	2	✓

	Input	Expected	Got	
✓	ABCD ABCD	4	4	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ 2-DP-Playing with chessboard

Jump to...

4-DP-Longest non-decreasing Subsequence ►

Question **1**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int longestNonDecreasingSubsequence(int sequence[], int n) {
    // Create a DP array
    int dp[n];

    // Initialize the dp array
    for (int i = 0; i < n; i++) {
        dp[i] = 1; // Each element is a non-decreasing subsequence of length 1
    }

    // Fill the DP array
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (sequence[i] >= sequence[j]) { // Non-decreasing condition
                dp[i] = dp[i] > dp[j] + 1 ? dp[i] : dp[j] + 1;
            }
        }
    }

    // Find the maximum value in the dp array
    int maxLength = 0;
    for (int i = 0; i < n; i++) {
        if (dp[i] > maxLength) {
            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 result = longestNonDecreasingSubsequence(sequence, n);
    printf("%d\n", result);

    return 0;
}

```

	Input	Expected	Got	
✓	9 -1 3 4 5 2 2 2 2 3	6	6	✓
✓	7 1 2 2 4 5 7 6	6	6	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ 3-DP-Longest Common Subsequence

Jump to...

1-DP-Playing with Numbers ►