Started on	Tuesday, 22 October 2024, 1:52 PM
State	Finished
Completed on	Tuesday, 22 October 2024, 2:00 PM
Time taken	7 mins 29 secs
Grade	**10.00** out of 10.00 (**100**%)

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>

// Function to calculate the number of ways to represent 'n' using 1 and 3
long long int countWays(int n) {
    // Create a dp array to store results for subproblems
    long long int dp[n + 1];

    // Initialize dp[0] = 1 as the base case
    dp[0] = 1;

    // Initialize dp values for all other positions
    for (int i = 1; i <= n; i++) {
        dp[i] = 0;
    }

    // Fill dp array using the recurrence relation
    for (int i = 1; i <= n; i++) {
        dp[i] += dp[i - 1];  // If we add 1
        if (i >= 3) {
            dp[i] += dp[i - 3];  // If we add 3
        }
    }

    // Return the result stored in dp[n]
    return dp[n];
}
```

```
27
28 int main() {
29     int n;
30     // Take input for number 'n'
31     //printf("Enter a number: ");
32     scanf("%d", &n);
33
34     // Output the result
35     printf("%lld\n",countWays(n));
36
37     return 0;
38 }
39
```

	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 ►

Started on	Tuesday, 22 October 2024, 2:00 PM
State	Finished
Completed on	Tuesday, 22 October 2024, 2:02 PM
Time taken	2 mins 13 secs
Grade	**10.00** out of 10.00 (**100**%)

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 %)

```c
#include <stdio.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

int findMaxPath(int n, int chessboard[n][n]) {
    // Create a DP table to store the maximum monetary value at each cell
    int dp[n][n];

    // Initialize the DP table
    dp[0][0] = chessboard[0][0];

    // Fill the first row (can only move right)
    for (int j = 1; j < n; j++) {
        dp[0][j] = dp[0][j - 1] + chessboard[0][j];
    }

    // Fill the first column (can only move down)
    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i - 1][0] + chessboard[i][0];
    }

    // Fill the rest of the dp table
    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            dp[i][j] = chessboard[i][j] + max(dp[i - 1][j], dp[i][j - 1]);
        }
    }

```

```
    // Return the maximum monetary value at the bottom-right corner
    return dp[n - 1][n - 1];
}

int main() {
    int n;
    // Input size of the chessboard
    //printf("Enter the size of the chessboard (n): ");
    scanf("%d", &n);

    int chessboard[n][n];

    // Input the monetary values of the chessboard
    //printf("Enter the chessboard values:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &chessboard[i][j]);
        }
    }

    // Output the maximum monetary value of the path
    printf("%d\n", findMaxPath(n, chessboard));
```

	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 ►

Started on	Tuesday, 22 October 2024, 2:03 PM
State	Finished
Completed on	Tuesday, 22 October 2024, 2:05 PM
Time taken	2 mins 24 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

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 max(int a, int b) {
    return (a > b) ? a : b;
}

int lcs(char* s1, char* s2, int m, int n) {
    // Create a dp table to store lengths of longest common subsequence
    int dp[m + 1][n + 1];

    // Build the dp array from the bottom up
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0) {
                dp[i][j] = 0;  // Base case: one string is empty
            } else if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;  // Characters match
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);  // Take the max
            }
        }
    }

    // The length of the LCS will be stored in dp[m][n]
    return dp[m][n];
}

int main() {
    char s1[100], s2[100];

    // Input two strings
    //printf("Enter first string: ");
    scanf("%s", s1);
```

```
35      //printf("Enter second string: ");
36      scanf("%s", s2);
37
38      int m = strlen(s1);
39      int n = strlen(s2);
40
41      // Output the length of the LCS
42      printf("%d\n", lcs(s1, s2, m, n));
43
44      return 0;
45  }
46
```

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	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 ►

Started on	Tuesday, 22 October 2024, 2:05 PM
State	Finished
Completed on	Tuesday, 22 October 2024, 2:08 PM
Time taken	2 mins 39 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

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>

// Function to find the length of the longest non-decreasing subsequence
int lnds(int arr[], int n) {
    int dp[n];

    // Initialize all dp[] values to 1
    for (int i = 0; i < n; i++) {
        dp[i] = 1;
    }

    // Compute length of LNDS using the DP approach
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (arr[i] >= arr[j]) {
                dp[i] = dp[i] > dp[j] + 1 ? dp[i] : dp[j] + 1;
            }
        }
    }

    // Find the maximum value in dp[] array
    int max_len = 0;
    for (int i = 0; i < n; i++) {
        if (dp[i] > max_len) {
            max_len = dp[i];
        }
    }

    return max_len;
}

int main() {
    int n;

    // Input the size of the sequence
    //printf("Enter the size of the sequence: ");
    scanf("%d", &n);

    int arr[n];

    // Input the sequence of numbers
    //printf("Enter the sequence: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Output the length of the longest non-decreasing subsequence
    printf("%d\n", lnds(arr, n));

    return 0;
}
```

```
52 |
```

	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-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity ►