2-DP-Playing with chessboard

Aim:

**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

Algorithm:

1. Read the size of the grid n and populate the n x n matrix board with user input.
2. Initialize a dp array with dp[0][0] = board[0][0] as the starting cell value.
3. Fill the first row of dp by cumulatively adding values from board.
4. Fill the first column of dp similarly by cumulatively adding values from board.
5. Use a nested loop to compute dp[i][j] = board[i][j] + max(dp[i-1][j], dp[i][j-1]) for all remaining cells.
6. Return dp[n-1][n-1] as the maximum monetary path value from the top-left to the bottom-right corner.
7. Print the result and end the program.

Code:

#include <stdio.h>

int max(int a, int b) {

return (a > b) ? a : b;

}

int maxMonetaryPath(int n, int board[n][n]) {

int dp[n][n];

dp[0][0] = board[0][0];

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

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

}

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

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

}

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

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

dp[i][j] = board[i][j] + max(dp[i - 1][j], dp[i][j - 1]);

}

}

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

}

int main() {

int n;

scanf("%d", &n);

int board[n][n];

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

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

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

}

}

int result = maxMonetaryPath(n, board);

printf("%d\n", result);

}

Output:

|  | **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.

Result:

The expected output was obtained