***Q1.*** ***You are given a 2D grid of integers (3x3) representing a simple board game. Each cell in the grid contains a positive integer representing the score you can obtain when you step on that cell. The goal is to find the maximum score you can achieve by starting from the top-left corner and moving to the bottom-right corner.***

***NOTE : Moves allowed are only moving right or down in the grid.***

***4 2 1***

***6 8 2***

***7 3 6***

***Possibilities :***

***4 + 2 + 1+ 2+ 6 = 15***

***4 + 2 + 8+ 2+ 6 = 22***

***4 + 2 + 8+ 3+ 6 = 23***

***4 + 6 +8+ 2+ 6= 26***

***4 + 6 + 8+ 3+ 6= 27***

***4 + 6 +7+ 3+ 6= 26***

***Maximum should be => 4 + 6 + 8+ 3+ 6 = 27***

***Solution:***

#include <iostream>

#include <algorithm>

const int ROWS = 3;

const int COLS = 3;

// Function to find the maximum score

int findMaxScore(int grid[ROWS][COLS]) {

int dp[ROWS][COLS];

// Initialize the first cell

dp[0][0] = grid[0][0];

// Initialize the first row

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

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

}

// Initialize the first column

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

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

}

// Fill the DP table

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

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

dp[i][j] = grid[i][j] + std::max(dp[i - 1][j], dp[i][j - 1]);

}

}

// The bottom-right cell contains the maximum score

return dp[ROWS - 1][COLS - 1];

}

int main() {

int grid[ROWS][COLS] = {

{4, 2, 1},

{6, 8, 2},

{7, 3, 6}

};

// Find and display the maximum score

int maxScore = findMaxScore(grid);

std::cout << "Maximum score: " << maxScore << std::endl;

return 0;

}