There are 3 questions for a total of 10 points.

1. There is an \( n \times n \) grid of one-way street network. At any intersection, you may either travel from east to west west to east or north to south. You want to compute the number of different ways in which you can travel from the north-west corner to the south-east corner. We will develop a dynamic programming solution for this problem.

![Example grid](image)

Figure 1: Example \( n = 6 \). In how many ways can you go from the top-left corner to the bottom-right corner?

Let \( W(i, j) \) denote the number of different ways of going from top-left corner to bottom-right corner for a grid of size \( i \times j \) (i.e., the grid has \( i \) horizontal lines and \( j \) vertical lines).

(a) (1/2 point) What is the value of \( W(1, j) \) for any \( j \) and \( W(i, 1) \) for any \( i \)?

(b) (1 point) Write \( W(i, j) \) in terms of \( W(i - 1, j) \) and \( W(i, j - 1) \) when \( i, j > 1 \). Give brief explanation for the relationship that your give.

(c) (1 point) Use the recursive formulation developed in (b) to design an algorithm that outputs the number of different ways of going from top-left corner to bottom-right corner of an \( n \times n \) grid. Discuss running time of your algorithm.

2. You are given \( n \) types of coin denominations of values \( v_1 < v_2 < \ldots < v_n \) (all integers). Assume \( v_1 = 1 \), so you can always make change for any integer amount of money \( C \). You want to make change for \( C \) amount of money with as few coins as possible.

We will solve this using Dynamic Programming. Let \( M(i, c) \) denote the minimum number of coins needed to make a change for \( c \) when we are allowed only coins of values \( v_1, v_2, \ldots, v_i \).

(a) (1/2 point) What is the value of \( M(1, c) \) for any positive integer \( c \)?

(b) (1 point) Give a recursive formulation for \( M(i, c) \). Give brief explanation as to why the recursive relationship that you give should hold.

(Try writing \( M(i, c) \) in terms of \( M(i - 1, c) \) and \( M(i, c') \) for some appropriate \( c' < c \))

(c) (1 point) Use the recursive formulation developed in (b) to design an algorithm that outputs the minimum number of coins needed to make a change for \( C \).

(d) (1 point) Modify your algorithm in (c) so that it outputs the number of coins of each value that is needed to make change for \( C \) using the minimum number of coins.

3. A string is called palindromic if it reads the same whether read left to right or right to left. Given a string \( S = s[1]s[2]\ldots s[n] \), you want to find a longest subsequence of \( S \) that is palindromic. For example, if \( S = \text{MAHATMA} \), then the longest subsequence of \( S \) that is palindromic is \( \text{MAHAM} \).

(a) (2 points) Give an algorithm that outputs the length of the longest subsequence that is a palindrome for a given input string. Discuss running time.
(b) (2 points) Give an algorithm that outputs a longest subsequence that is a palindrome for a given input string. Discuss running time.