Weekly Assignment 9:
Dynamic Programming 1

1. Run the dynamic programming algorithm for the knapsack problem on the following input and capacity $C = 5$:

Item	Value	Size
1	8	3
2	3	2
3	9	4
4	6	1

   Show both the complete matrix that is filled to compute the maximum possible total value, and the steps taken by the algorithm to reconstruct the corresponding collection of items.

2. Write pseudocode for a bottom-up implementation of the dynamic programming algorithm for computing the best order in which to multiply a sequence of matrices.

3. You are given an n-by-n grid, where each square (i, j) contains $c(i, j)$ gold coins. Assume that $c(i, j) \geq 0$ for all squares. You must start in the upper-left corner and end in the lower-right corner, and at each step you can only travel one square down or right. When you visit any square, including your starting or ending square, you may collect all of the coins on that square. Give an algorithm to find the maximum number of coins you can collect if you follow the optimal path. Discuss the correctness and time complexity of your algorithm.

4. You are given a string of n characters $s = s[1], ..., s[n]$, which you believe to be a corrupted text document in which all spaces and punctuation have vanished (so that it looks something like "itwasthebestoftimes..."). You wish to reconstruct the sequence of words in the document using a dictionary, which is given to you as a Boolean function $d(w)$ that takes a string w as input and returns `true` if w is a valid word and `false` otherwise. Give a dynamic programming algorithm that determines whether the string s can be reconstituted as a sequence of valid words. Describe the recurrence equations on which your algorithm is based, explain why these equations hold, and analyze the time complexity of your algorithm. You may assume that calls to $d(w)$ take constant time.