Name(s)_________________________________ _____________________________________

Algorithms, Assignment 06: Dynamic Programming

1. Consider the Two-Dimensional Jump-It problem from the assignment description.

    a. Describe, in plain English, how the Two-Dimensional Jump-It problem exhibits overlapping subproblems.

    b. Describe, in plain English, how the Two-Dimensional Jump-It problem exhibits optimal substructure.

Name(s)_____________________________________ ____

2. Consider the Knapsack Problem and these values:

	ITEM 1	ITEM 2	ITEM 3	ITEM 4
WEIGHT	10	20	35	50
VALUE	60	70	120	180

a. We discussed the 0-1 Knapsack Problem in class, but there is also a version known as Fractional Knapsack. In Fractional Knapsack, you are allowed to break items into any fractional amount.

   i. Describe an optimal greedy algorithm for Fractional Knapsack.

   ii. Provide the output of your greedy algorithm for the table above when the Knapsack capacity is 55.

b. Provide the output (optimal value, not the table) of our dynamic programming algorithm for the 0-1 Knapsack Problem and the table above when the Knapsack capacity is 55.

Name(s)____ ____

c. Provide a proof of correctness for the 0-1 Knapsack Algorithm that we discussed in class.

Name(s)____________________________________ ____

3. In the previous assignment we discussed a solution to the problem of handing back change in US coinage. A greedy solution to that problem worked because the US coin system is "canonical." Now consider a non-canonical change system with coins of values.

   For example, if the coins are from the set {1, 5, 6, 9} and the change to hand back is 11, our greedy solution would return one 9 and two 1s instead of the optimal solution of one 5 and one 6.

   For this problem you will give a dynamic programming algorithm for making change using any coin system with values: $v_1 < v_2 < \ldots < v_n$ (all integers), where $v_1 = 1$.

   a. Describe a table to be filled in by a dynamic programming solution to this problem. What are the values in the table and what are its dimensions?

   b. Write a recursive definition for the function that defines the values in the table.

   c. What is the running time of filling in the table?

   d. Show the table if v_1=1, v_2=5, and v_3=6, and the amount of change to return is 10.

Name(s)____________________________________ ____

4. *Arbitrage* is a money-making scheme involving anomalies in international currency exchange rates. For example, imagine that 1 U.S. dollar buys 0.8 Zambian kwachas, 1 Zambian kwacha buys 10 Mongolian tughriks, and 1 Mongolian tughrik buys 0.15 U.S. dollars.

   A trader can start with 1 U.S. dollar and buy $0.8 \times 10 \times 0.15 = 1.2$ U.S. dollars. Large amounts of money can be made by capitalizing on such anomalies before they're detected and corrected by the markets.

   Assume that we're given n currencies $c_1, \ldots, c_n$ and the exchange rate between every pair of currencies; that is, c_i buys $R[i, j]$ units of currency c_j. Also assume that there are no cycles that enable you to get arbitrarily rich.

   Objective: describe an algorithm for computing the maximum amount of currency that you can obtain by starting with 1 unit of that currency (for all currencies). For full credit, make sure that your algorithm is as fast as possible.

   Hint: $\log_b(c_i \cdot c_{i+1} \cdot \ldots \cdot c_j) = \log_b c_i + \log_b c_{i+1} + \cdots + \log_b c_j$
   (Summing the logs of numbers is equivalent to the log of the product of those same numbers.)