Name: _____

Entry number: _____

There are 5 questions for a total of 75 points.

---

1. (5 points) You are given n items and a sack that can hold at most W units of weight. The weight of the i^{th} item is denoted by $w(i)$ and the value of this item is denoted by $v(i)$. The items are indivisible. This means that you cannot take a fraction of any item. Design an algorithm that determines the items that should be filled in the sack such that the total value of items in the sack is maximized with the constraint that the combined weight of the items in the sack is at most W. You may assume that all the quantities in this problem are integers. Give pseudocode, discuss running time, and argue correctness.

2. (15 points) You are given n types of coin denominations of values $v_1 < v_2 < ... < v_n$ (all integers). Assume $v_1 = 1$, so you can always make change for any integer amount of money C. Design an algorithm that makes change for an integer amount of money C with as few coins as possible. Give pseudocode, discuss running time, and argue correctness.

3. (15 points) Recall, the problem of finding a minimum vertex cover of a tree. Suppose, you are given a rooted tree T with root r. For every node v, let $C(v)$ denotes the set of children of the node v in T. So, for a leaf node v, $C(v) = \{\}$. We will try to find the minimum vertex cover using Dynamic Programming. For any node v, let $M(v, 0)$ denote the size of the vertex cover S of the subtree rooted at v such that $v \notin S$ and S is the smallest such cover. Let $M(v, 1)$ denote the size of the vertex cover S of the subtree rooted at v such that $v \in S$ and S is the smallest such cover. So, the size of the minimum vertex cover of the given tree is $\min\{M(r, 0), M(r, 1)\}$. Give the recursive formulation for $M(., .)$ and use this to give an algorithm for finding the size of minimum vertex cover. Your algorithm should also output a minimum vertex cover of T. Give pseudocode, discuss running time, and argue correctness.

4. (20 points) You are given a rectangular piece of cloth with dimensions $X \times Y$, where X and Y are positive integers, and a list of n products that can be made using the cloth. For each product $i \in \{1, ..., n\}$ you know that a rectangle of cloth of dimensions $a_i \times b_i$ is needed and that the final selling price of the product is c_i. Assume the a_i, b_i, and c_i are all positive integers. You have a machine that can cut any rectangular piece of cloth into two pieces either horizontally or vertically. Design an algorithm that determines the best return on the $X \times Y$ piece of cloth, that is, a strategy for cutting the cloth so that the products made from the resulting pieces give the maximum sum of selling prices. You are free to make as many copies of a given product as you wish, or none if desired. Give pseudocode, discuss running time, and argue correctness.

5. (20 points) Consider the 0-1 Knapsack problem that we discussed in lecture. Given n positive integers $w_1, ..., w_n$ and another positive integer W, the goal is to find a subset S of indices such that $\sum_{i \in S} w_i$ is maximised subject to $\sum_{i \in S} w_i \leq W$. We studied a dynamic program for this problem that has a running time $O(n \cdot W)$. We know that this is bad when the integers are very large. The goal in this task is to improve the running time at the cost of optimality of the solution. Let us make this more precise. Let OPT denote the value of the optimal solution. For any given $\varepsilon \in (0, 1]$, design an algorithm that outputs a solution that has value at least $\frac{OPT}{(1+\varepsilon)}$. The running time of your algorithm should be of the form $O\left(\frac{f(n)}{\varepsilon}\right)$, where $f(n)$ is some polynomial in n. You need not give the best algorithm that exists for this question. As long as $f(n)$ in the running time is polynomial in n, you will get the full credit.

   (Hint: Try modifying the dynamic programming solution for the knapsack problem.)