**CS575 Homework 4**

**Submit a scanned pdf file through blackboard by 11:59pm on May 4. No late submission will be accepted.**

**Your Name**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Section:** \_\_\_\_\_\_\_\_\_\_\_\_

I promise to follow the academic honesty requirements of the Binghamton University. I agree that I will fill out and sign an official form that I have cheated if I get caught cheating. I understand that this form will be stored by the university. Furthermore, I understand that the minimum penalty for cheating is getting a grade of 0 for this assignment and my letter grade will be decreased by one level.

**Sign:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. [20%] You are given a **0-1** **knapsack** problem where the capacity of the knapsack W = 35 and the set of items S = {(i1, 5, $50), (i2, 25, $140), (i3, 10, $60)} where each element in set S is a tuple of (item ID, weight, profit).Solve the given 0-1 knapsack problem using the dynamic programming method discussed in Chapter 12. Clearly show every step.
2. [20%] A set {3, 4, 5, 6} is given. For the set, find **every subset** that sums to S = 15.
   1. [10%] Solve the problem using the depth first method. Draw a state space tree and clearly show every step. Also, number the nodes in the sequence of visiting them.
   2. [30%] Find the subsets via backtracking. Draw a (pruned) state space tree and clearly show every step. Number the nodes in the sequence of visiting them too.
3. [20%] When the capacity of the knapsack is 12, solve the following **0-1 knapsack** problem using the backtracking algorithm discussed in Chapter 13.

*i pi wi pi* / *wi*

1 $10 5 $2

2 $30 5 $6

3 $40 2 $20

4 $50 10 $5

1. [20%] Assume that a hash table has 17 buckets where each bucket has only one slot. A simple hash function: home bucket = key % 17 (where % is a mod function) is used to compute the home bucket based on the key. You are supposed to insert the following keys to the hash table: 6, 15, 34, 29, 28, 11, 21, 7, 0, 32, 30, 45 using the following overflow handling methods.
2. Use the linear probing (linear open addressing) method to handle overflows, if any.
3. Use the sorted chaining method to deal with overflows, if any.
4. [20%] Somebody takes a naïve, brute-force approach to computing a parallel prefix sum. Given a list of *n* elements (x0, x1,…., xn-1), he computes the parallel prefix sum using *n* processors where processor *i* (0 ≤ *i* ≤ *n*-1) computes *yi* as follows:

*y*0 = *x*0

*y*1 = *x*0 + *x*1

*y*2 = *x*0 + *x*1+ *x*2

*…*

*y*n-1 = *x*0 + *x*1+ … + *x*n-1

What are the span and work of this algorithm? Briefly explain why you get the specific span and work.