1. Hash Table (10 points) Exercises 11.2-1, page 261

2. **Hash Function** (40 points) Consider inserting keys 3,4,2,5,1 in the order given into a hash table of length $m = 5$ using hash function $h(k) = k^2 \bmod m$ (k^2 is the auxiliary function).

(1) Using $h(k)$ as the hash function, illustrate the result of inserting these keys using chaining. Also, compute the load factor α for the hash table resulting from the insertions.

(2) Using $h(k)$ as the primary hash function, illustrate the result of inserting these keys using open addressing with linear probing.

(3) Using $h(k)$ as the primary hash function, illustrate the result of inserting these keys using open addressing with quadratic probing, where $c1=1$ and $c2=2$.

(4) What different values can the hash function $h(k) = k^2 \bmod m$ produce when $m = 11$? Carefully justify your answer in detail.

3. BST: Using the definitions on p. 1177 of our textbook for *depth* of a tree node and *height* of a tree, consider the set of keys **K** = < 10, 4, 2, 8, 7 > and the different possible insertion orders for the keys in **K**. Based on the different possible insertion orders and their resulting Binary Search Trees, answer the following questions. (10 points)

a) What is the minimum height of a Binary Search Tree constructed from **K**? Show an insertion order for the keys in **K** that generates a Binary Search Tree of minimum height. Draw the corresponding Binary Search Tree.

b) Are there any other insertion orders (beyond what you found in (a) above) for the keys in **K** that produce a Binary Search Tree of minimum height? If so, provide one such sample insertion order and its accompanying Binary Search Tree.

c) What is the maximum height of a Binary Search Tree constructed from **K**? Show an insertion order for the keys in **K** that generates a Binary Search Tree of maximum height. Draw the corresponding Binary Search Tree.

4. BST: Exercise 12.2-5 on p. 293 in textbook. (25 points)

5. Algorithm Design (20 points).

a. Lowest Common Ancestor. Given the values of two nodes in a binary search tree, find the lowest (nearest) common ancestor. You may assume both values already exist in the tree.

b. Union and Intersection of two Linked Lists. Given two Linked Lists, design an algorithm to efficiently create union and intersection lists that contain union and intersection of the elements present in the given lists. Order of elements in output lists doesn't matter.