1. Hash Table (20 points) Exercises 11.2-1, page 261

For each pair of distinct keys k_i and k_j, define an indicator RV x_{ij}, $1 \le i < j \le n$. Thus, $x_{ij} = 1$ if keys k_i and k_j collide and 0 otherwise. If X denotes the RV that gives the total number of collisions, we have

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}.$$

We want to compute $\mathrm{E}[X]$. By linearity of expectation, we have

$$\mathrm{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathrm{E}[x_{ij}]. \tag{1}$$

We compute $\mathrm{E}[x_{ij}] = \Pr\{x_{ij} = 1\}$ as follows. By the simple uniform hashing assumption, for any index p, the probability that both k_i and k_j hash to the value p is equal to $1/m \times 1/m = 1/m^2$. Since p can take on m different values, the probability that k_i and k_j collide is $m \times 1/m^2 = 1/m$. In other words, $\mathrm{E}[x_{ij}] = \Pr\{x_{ij} = 1\} = 1/m$. Since the number of indicator RVs is $n(n-1)/2$, we have from Equation (1), the expected number of collisions $\mathrm{E}[X] = n(n-1)/(2m)$.

2. **Hash Function** (60 points, 15 for each sub-question) 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.

{3, 4, 2, 5, 1} gets mapped to {4, 1, 4, 0, 1}.
They are loaded in the table that now looks like this:
[0] --> 5
[1] --> 1,4
[2] --> NIL
[3] --> NIL
[4] --> 2,3

Load factor: n / m = 5 / 5 = 1.

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

{3, 4, 2, 5, 1} --> {4, 1, 4, 0, 1}
Length m = 5.

h(k,i) = (k^2 + i) % 5

h(3,0) = 3^2%5 = 4
h(4,0) = 4^2%5 = 1
h(2,0) = 2^2%5 = 4, collision
h(2,1) = 2^2 + 1 % 5 = 0
h(5,0) = 5^2 + 0 % 5 = 0, collision
h(5,1) = 5^2 + 1 % 5 = 1, collision
h(5,2) = 5^2 + 2 % 5 = 2
h(1,0) = 1^2 + 0 % 5 = 1, collision
h(1,1) = 1+1 % 5 = 2, collision
h(1,2) = 1+2 % 5 = 3

Final mapping is {4, 1, 0, 2, 3}

m	k
1	4
2	5
3	1
4	3
0	2

(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.

h(k,i) = (k^2 + i + 2*i^2) % m

h(3,0) = 4
h(4,0) = 1
h(2,0) = 4, collision
h(2,1) = (2^2 + 1 + 2*1^2)%5 = (4+1+2)%5 = 7%5 = 2
h(5,0) = 0
h(1,0) = 1, collision ...
h(1,1) = (1 + 1 + 2*1^2)%5 = (4%5) = 4, collision ...
h(1,2) = (1 + 2 + 2*2^2)%5 = (11%5) = 1, collision ...
h(1,3) = (1 + 3 + 2*3^2)%5 = (1+3+18)%5 = 2, collision ...
h(1,4) = (1 + 4 + 2*4^2)%5 = (1+4+32)%5 = 2, collision ...

Do we have a bad choice of c1 and c2 here, such that
h(1,i) never finds a suitable slot?
Mapping is otherwise {4, 1, 2, 0, X}

m	k
1	4
2	2
4	3
0	5

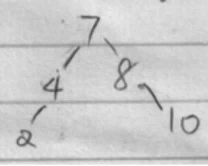
(4) What different values can the hash function $h(k) = k^2 \bmod m$ produce when $m = 11$? Carefully justify your answer in detail.

The hash function can produce 6 different values: 0,1,3,4,5,9.
For any integer k, it can be written as k = 11n + m, m = 0,1,2,...,10.
Consider k^2, $k^2 = (11n)^2 + 22nm + m^2$. Obviously remainder can only come from m^2. Since m = 0,1,2,...,10, from m^2 mod 11, we can only get possible remainders 0,1,3,4,5,9.

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. (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.

(a) minimum height : 2



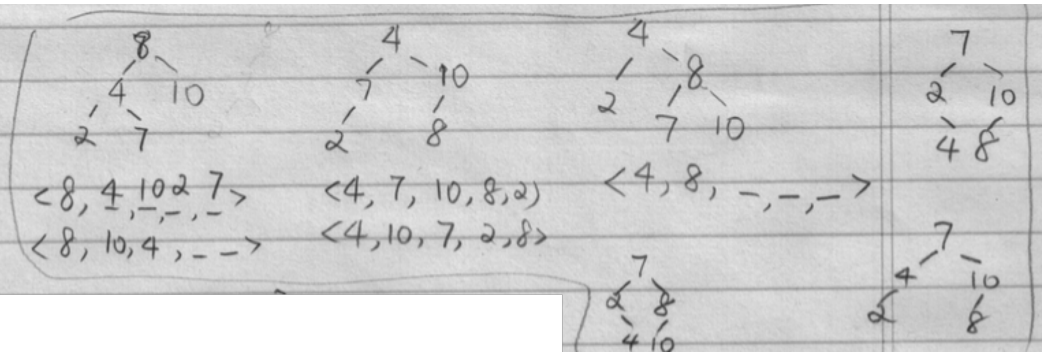
when insert <7, 4, 8, 10, 2>

first

order doesn't matter

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.

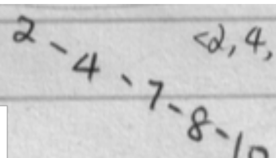
(b)



<8, 4, 10, 2, 7>
<8, 10, 4, _, _>
<4, 7, 10, 8, 2>
<4, 10, 7, 2, 8>
<4, 8, _, _, _>

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.

(C) maximum height: 4



<2, 4, 7, 8, 10>

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

Let x be a node with two children. In an inorder tree walk, the nodes in x's left subtree immediately precede x and the nodes in x's right subtree immediately follow x. Thus, x's predecessor is in its left subtree, and its successor is in its right subtree.

Let s be x's successor. Then s cannot have a left child, for a left child of s would come between x and s in the inorder walk. (It's after x because it's in x's right subtree, and it's before s because it's in s's left subtree.) If any node were to come between x and s in an inorder walk, then s would not be x's successor, as we had supposed.

Symmetrically, x's predecessor has no right child.

4. 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.