CSCE 221 Cover Page
Homework Assignment #3
Due November 30 to CSNet

First Name Kevin Last Name Chou UIN 224000120

User Name k11 E-mail address k11@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources			
People			
Web pages (provide URL)			
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name Kevin Chou Date 11/30/15

Homework 3 due November 30

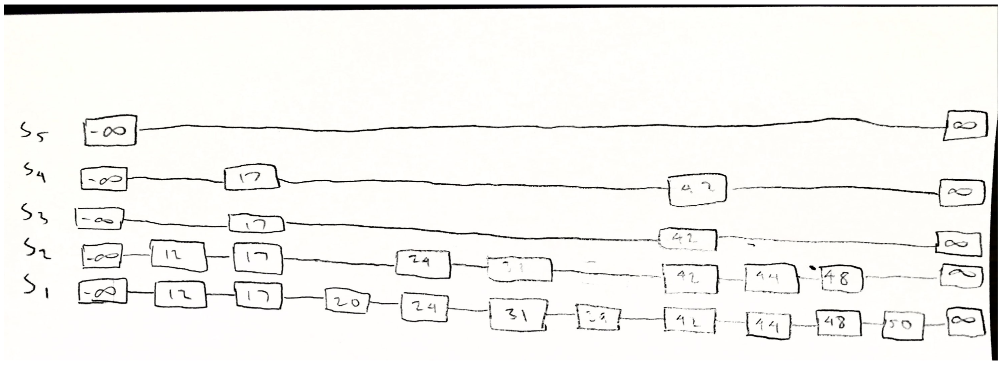
1. **(10 points) R-10.17 p. 493**
   For the following statements about red-black trees, provide a justification for each true statement and a counterexample for each false one.

   a. A subtree of a red-black tree is itself a red-black tree.
      False, if a subtree is rooted at a node that is red, then it does not follow the rule that a red-black tree must have a black root.
   b. The sibling of an external node is either external or it is red.
      True, a red-black tree leaves have the same black depth and if an external node has an internal sibling that is black, then the 2 siblings won't have the same black depth.
   c. There is a unique (2,4) tree associated with a given red-black tree.
      False, there are different red-black trees that can be represented in the same (2,4) tree.
   d. There is a unique red-black tree associated with a given (2,4) tree.
      False, a (2,4) tree can be represented by multiple different red-black trees.

2. **(10 points) R-10.19 p. 493**
   Consider a tree T storing 100,000 entries. What is the worst-case height of T in the following cases?

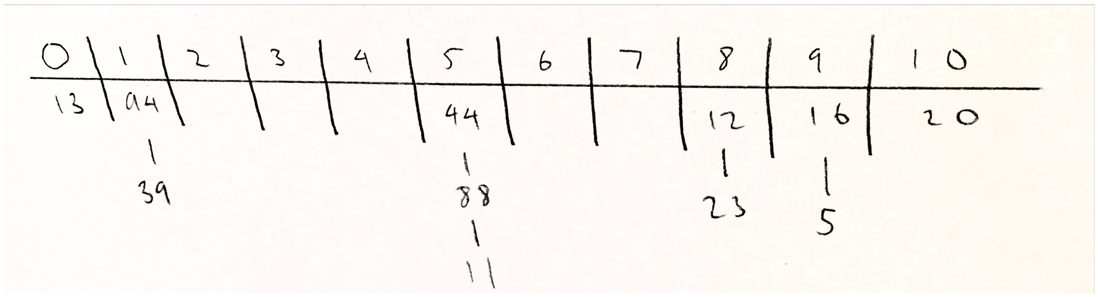
   a. T is an AVL tree.
      1.44log100,000
   b. T is a (2,4) tree.
      Log100,000
   c. T is a red-black tree.
      2log100,000
   d. T is a binary search tree.
      100,000

3. **(10 points) R-9.16 p. 418**
   Draw an example skip list that results from performing the following series of operations on the skip list shown in Figure 9.12: erase(38), insert(48,x), insert(24,y), erase(55). Record your coin flips, as well.


S_5 $-\infty$ ∞
S_4 $-\infty$ 17 42 ∞
S_3 $-\infty$ 17 42 ∞
S_2 $-\infty$ 12 17 24 31 42 44 48 ∞
S_1 $-\infty$ 12 17 20 24 31 39 42 44 48 50 ∞

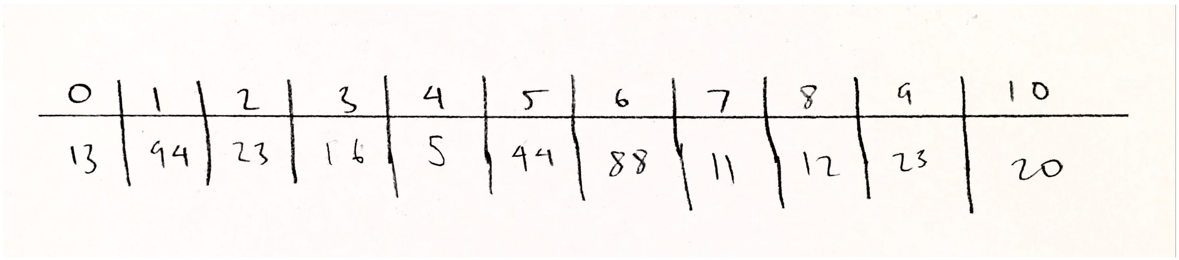
Both insert 48 and 24 were 2 heads.

4. (10 points) R-9.7 p. 417
Draw the 11-entry hash table that results from using the has function, h(k) = (3k + 5) mod 11, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.



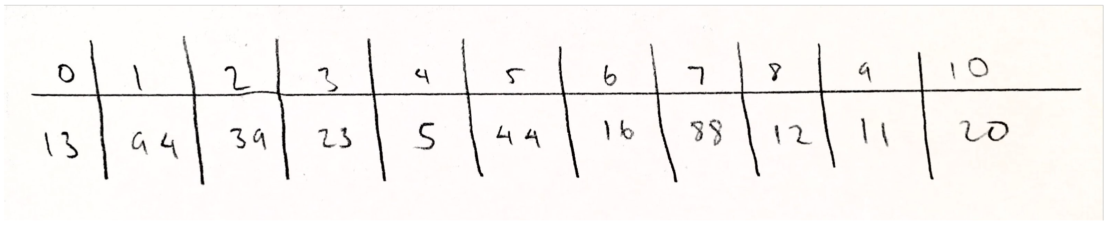
0	1	2	3	4	5	6	7	8	9	10
13	94				44			12	16	20
	39				88			23	5	
					11					

5. (10 points) R-9.8 p. 417
What is the result of the previous exercise, assuming collisions are handled by linear probing?



0	1	2	3	4	5	6	7	8	9	10
13	94	23	16	5	44	88	11	12	23	20

6. **(10 points) R-9.10 p. 417**
What is the result of Exercise R-9.7, when collisions are handled by double hashing using the secondary hash function $h_s(k) = 7 - (k \bmod 7)$?



0	1	2	3	4	5	6	7	8	9	10
13	94	39	23	5	44	16	88	12	11	20

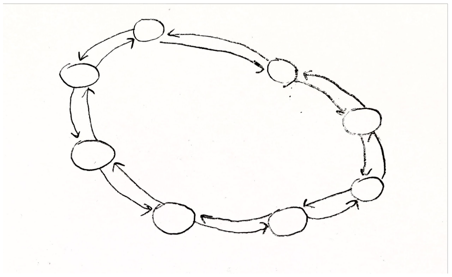
7. **(10 points) R-8.2 p. 361**
How long would it take to remove ⌈log n⌉ smallest elements from a heap that contains n entries using the removeMin() operation?

$O(\log^2 n)$

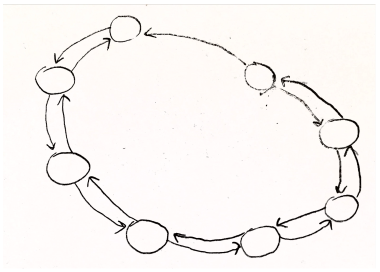
8. **(10 points) R-8.7 p. 361**
An airport is developing a computer simulation of air-traffic control that handles events such as landings and takeoffs. Each event has a time-stamp that denotes the time when the event occurs. The simulation program needs to efficiently perform the following two fundamental operations: • Insert an event with a given time-stamp (that is, add a future event)• Extract the event with smallest time-stamp (that is, determine the next event to process) Which data structure should be used for the above operations? Why?

Heap because insertion and remove operations would both have runtime O(logn).

9. **(10 points) R-13-3 and R-13-4, p. 654**



10. (10 points) R-13.8 p.655



11. (10 points) R-13.16, p. 656

a.) Adjacency list because an adjacency matrix use up a lot of space. It would allocate 100,000 edges for a graph that only has 20,000 edges.

b.) Adjacency matrix because the adjacency list would use up more space. It would allocate 20,000,000 edges but only 100,000 are needed.

c.) Adjacency matrix because you would have to iterate through the entire list to check which could take much longer than a matrix.

12. (10 points) R-13.17, p. 656

Prim's algorithm

3 -> 5 (115)   Total weight = 115+120+155+160+170+175+180 = 1075

1 -> 8 (120)

2 -> 9 (155)

4 -> 5 (160)

5 -> 8 (170)

6 -> 7 (175)

2 -> 6 (180)

13. (10 points)

You want to help CS/CSE freshman students to prepare their course schedules for the first two years in the lower level division. By building a directed graph suggest order in which they should schedule their courses taking into account their corresponding prerequisites. A set of vertices represents courses and a set of edges represents a dependence of a given course on a course prerequisite.

CSCE 111 -> CSCE 121 -> CSCE 221/222 -> CSCE 312/314

CHEM 101/111 -> CHEM 102/112

MATH 151 -> MATH 152 -> MATH 251 -> MATH 304

14. (10 points) R-13.31, p. 656

It will look like a spanning tree. Each path is searched before going back to the root and going down another path to search.

15. (10 points)

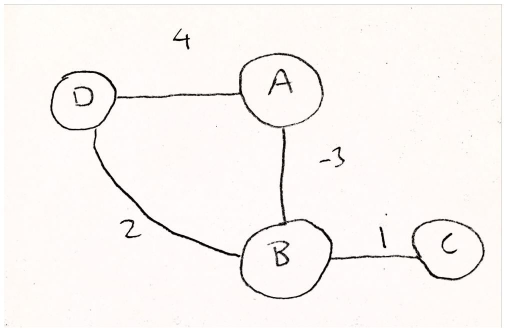
Write what the running time, and provide its justification, of the Dijkstra's algorithm is for a sparse and dense graph and the priority queue implemented based on

a. a binary heap
   Sparse graph: $O((V+E)\log V)$
   Dense graph: $O(V^2\log V)$
b. an unsorted list
   Sparse graph: $O(V^2+E)$
   Dense graph: $O(V^2)$
c. a sorted list
   Sparse graph: $O(V^2+E)$
   Dense graph: $O(V^2)$

16. (10 points) C-13.10, p. 658

Use a DFS traversal.

17. (10 points) C-13.15, p. 659



18. (10 points) C-13.18, p. 659

Implement a relax vertices function.

```
relaxVertice(a,b,c):

if (d[b] < d[a] + c(a,b)) {

    d[b] = d[a] + c(a,b);

    p[b];
```

Create a maximum priority queue and initialize the first vertex to zero and vertices in the queue to negative infinity and then find the max and relax its adjacent vertices. Do this until the queue is empty. Using a binary heap to implement the maximum priority queue, the time complexity of the algorithm would be O((V+E)logV).