Practice Midterm 2

This test has 10 questions worth a total of 120 points, and is to be completed in 110 minutes. The exam is closed book, except that you are allowed to use one double sided written cheat sheet (front and back). No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Write the statement out below in the blank provided and sign. You may do this before the exam begins.

#	Points	#	Points
1	15	6	12
2	9	7	12
3	0	8	9
4	6	9	12
5	20	10	25
		Total	120

1. Dat-uhhh Structures: (15 pts) Provide the solutions in the blanks under the questions. Each question is worth 3 points.

a) How many nodes would be in the left subtree of a complete binary search tree where height (h) > 0 and the root starts at a height of 0. Note, if you want to use height in your calculations, you must use the height of the full binary search tree, not subtrees.

b) How many nodes would be in the right subtree of the right subtree in a full binary search tree. The same assumptions should be made as in the previous problem

c) How many different heaps can be both max heaps and min heaps? If there are none, justify why, tell us the cases that result in heaps that fit this criteria.

d) Can red-black trees have only black links? When is this the case?

e) What is wrong with the following code snippet? Assume that it compiles properly, there is a constructor, and a hashcode method that has a hashcode function works well (spreads everything well).

```
Public class Human{
int legs;
int arms;
….
@Override
public boolean equals(Human no) {
        return legs == no.legs && arms == no.arms;
}
```

2) Flash Union: (9 pts)

a) Beary Allen, a fast UC Berkeley student wants to make trips between cities. He wants to see if they are connected in the fastest way possible- as a result, he will use the fastest union data structure that we were taught. His *travel* method acts like *connect* or *union* in a normal union data structure. Draw the Union Structure in each box after all the commands have executed. (5 pts)

Gotham Metropolis Central City Star City Atlantis Coast City
travel(Gotham, Star City); travel(Atlantis, Metropolis); travel(Central City, Coast City);
travel(Metropolis, Central City);
travel(Gotham, Central City)

b) Beary Allen wants to determine the longest time that it would take for him to check if 2 cities are connected. Please write the runtime in terms of N, the number of cities. (2 pts)

c) Beary decided that the runtime is not fast enough for the fastest man in the world. Can he go faster by changing the way the Union Data Structure works? If so, briefly describe how you would do so. (1 pt)

3) Riddle me this: (0 pts)
Which president wears the biggest hat?

4) Pitching and Catching: (6 pts) What does the following code display? You need not use all the lines.

```
  public static void triedTooHard(String[] n){
     try{
        for(int i = 0; i < n.length + 1; i++){
           System.out.println(n[i].charAt(0));
        }}
     catch(NullPointerException E){
        System.out.println("Tried" + n.length + "hard");}
     catch(IndexOutOfBoundsException E){
        String[] t = new String[n.length+1];
        System.arraycopy(n, 0, t, 1, n.length);
        triedTooHard(t);
     }
     finally{
        System.out.println(n.length);}}

public static void main(String[] args){
  triedTooHard(new String[] {"my ", "everything ", "my", "eternal"});
}
```

5) B-arbor Day: (20 pts) Woody TreePecker is a bit sad because he wants a place to perch. The only problem is he doesn't have a nice bushy tree to perch in- all he has is spindly LinkedLists. He has employed you, professional tree barber (or gardner whatever you want to call it), to give him some really nice trees. Now you, being the conservationist you are, want to turn the LinkedLists into Balanced Binary Search Trees.

Write the code that will convert a sorted LinkedList (one from java's documentation) into a Balanced Binary Search Tree (that can contain generics). For consistency, your main method that converts the LinkedList to a Binary Search Tree should be labeled **LinkedListToBST**. You will be given the remainder of this page and one more page to complete your class.

6) Test making you sad? Call that A-simp-totics: (12 pts) Write the Asymptotic runtime of the following function. Use the tightest bounds possible. Each problem is worth 4 points.

1.
```
public static void louwTheWay(int N){
   for int(i = 0; i < N / 2 ; i ++){
      System.out.println("you");
    }
   louwTheWay(N - 1); louwTheWay(N-2);\\lie
}
```

2.
```
public static void takeCaretik(int N){
        for(int i = 0; i < (N*1000+30^100)/N ; i ++){
             System.out.println("I'll take care of you.");
             }
     takeCaretik(N/4);
  }
```

3.
```
public static void weDontTalk(int N){
    for(int i = 0 ; i < N; i++){
       for (int j = 0 ; j < i; j++){
        System.out.print("Anymore")
         }
     }
    weDontTalk(N/2) \\Like we used to
  }
```

7) MashedSet: (12 pts)

a) We have a HashSet, but a goon broke part of our code so it does not handle duplicates properly. This means that duplicates can added without care. Luckily, your hashcode, which is quite nice, is still intact (it was encrypted well), along with this, the spread of the hashcode is pretty good. Determine how long it would take in order to traverse your broken HashSet to find and delete ALL duplicates of ONE key. For example, how long would it take to find the key "macaroni". Provide the runtime (It is your choice what symbol to use) based off our prior assumptions. Give reasons for both runtimes. Correct runtimes with wrong explanations will be given 0 points.

b) You're trying to figure out how the goon managed to break your code so easily. You spent so many hours on it, and he managed to change it very quickly so that duplicates aren't handled. What is the most probable way he did this?

c) Pretend that you did not remove the duplicates from the faulty HashSet and you wanted to keep track of how many times each item was inside of the HashMap. How would one do this? We would like our solution to be as fast as possible.

8) Rotations and Flips: (9 pts)

a) What series of inserts would result in a perfectly balanced binary search tree for the following input: 1, 2, 3, 4, 5, 6 , 7 , 8 , 9 , 10, 11, 12, 13, 14, 15. Draw the resulting Binary Search Tree.

b) Delete the item 12 from the above Binary Search Tree and draw the results.

c) Insert 7 into the following red black tree. Show your steps using the red-black tree method for full credit. ½ credit will be given for converting into a 2-3 tree and back.

9) Runtimes not finished call that Not dun-times: (12 pts) Write all runtimes in Theta or Omega and O. Ignore constants but do not ignore lower order terms. Provide an explanation for your runtimes.

a) Runtime of putting N presorted keys into a binary search tree.

b)You have a minheap and you will perform deleteMin operations until the min-heap is empty. Whenever an element is deleted, it is added to a Binary Search Tree.

c) Inserting N elements into a HashMap that all have the same key and different values.

d) Given an array sorted from least to greatest put all the elements into a stack (lowest index to highest index)then pop off all the elements. Each time an element is popped off put it into a maxheap.

10) NBA: National Bongoola Association: (25 pts)

You are the coach of a prestigious Bongoola team. Your job as coach is to make sure that, at any given time, the best players on your team for each positions are on the field. Bongoola, being a great community team sport, invites all people of the community to join. Each player will be assigned a position that they can play- there are a total of **M** positions- and a unique team number. For this problem, we will use **N** to be the total amount of people on your team and **P** to be the people signed up for a certain position.

You have your starting **M** players, but as the game progresses, you will need to replace them. You will base your replacement of people using the *bodacious* factor. During the game, only 1 player's bodacious factor goes down, and once they are benched, their bodacious factor doesn't change. Every **Z** minutes, 1 player on the field will have their bodacious factor decrease and you will be able to replace the player if needed. Once the next eligible player's (a player who plays the same position) bodacious factor exceeds the current player, they are considered a better choice. You are guaranteed that only 1 of the players at a time will need to be replaced. Changing a player's bodacious factor should take $\boldsymbol{\Theta}$(l) time, replacing a player should take $\boldsymbol{\Theta}$(log(P)) time, and putting a new player in the current lineup should take $\boldsymbol{\Theta}$(1) time.

On the next page, write a detailed description of what data structures you will use to make the operations stated above run in the provided time. State any assumptions you need to regarding anything (no you cannot assume that everything is done by some magical warlock).