Data Structures Exam 2 (94%)

Data Structures (Temple University)

- You are allowed to clarify any answer you give.
  - All questions are essay questions, including the multiple choice.
- You are allowed to ask for clarification.
- Important `String` methods:
  - `length()`
  - `charAt(int index)`
  - `substring(int start, int end)`
  - `startsWith(String s)`
- `Tree` methods:
  - `add`
  - `remove`
  - `size`
  - `contains`
- `Map` methods:
  - `put`
  - `get`
  - `containsKey`
  - `containsValue`
- Things are never as complicated as they appear, especially the math.
- Never leave a question blank, even if you don't know the answer. We can't give partial credit to blanks.
- Extra credit is available for exceptional answers (up to five additional points).

Don't Panic

1 Data Structures

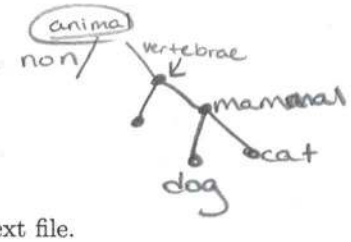
For each of the following questions, indicate which data structure would be best suited to handle each scenario. Your choices are `ArrayList`, `LinkedList`, `Stack`, `Queue`, `Tree`, `Set`, `Map` and `Priority Queue`. You may justify your answer with a single sentence.

1. (3 points) You need a data structure to simulate cars coming to and leaving from a particular traffic light.

   Queue, because the first car at the light will be the first to leave, and assuming it's one lane, the cars will leave in the exact order of arrival, which is a characteristic of a queue.

2. (3 points) You need to store different animals based on their taxonomy. For each animal, you classify it based on what features it has. For example, a cat would first be sorted by seeing if it had a vertebrae, then sorted based on the fact it was a mammal, and so on.

   Tree, because different animals can be placed as children of different classifications



3. (3 points) You want to count the frequency in which individual characters occur in a text file.

   Map would be best for this because a map can hold a key value (the character) and info regarding that key (frequency of the character).

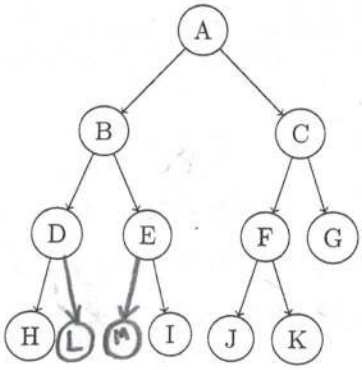
4. (3 points) In the card game Magic: The Gathering, players play cards to cast spells. Any single spell cast could potentially be reacted to by casting another spell in response, and most recent spell can itself be responded to by another spell, *ad nauseam*. These spells the resolve beginning with the most recent card cast, ending with the initial spell cast. What data structure should be used to store these spells so they resolve in the correct order?

   Stack, because a stack is LIFO, so the most recent card will be the first to be removed.

12

12 points

2 Tree Relations and Vocabulary



5. (10 points) Please fill in the blanks to describe the relationships in the above tree.

(a) Node C is a(n) grandparent of J

(b) Node I is a grandchild of B.

(c) All the nodes to the left of A compose a subtree of A's tree, with B as the root.

(d) Removing nodes H, I would make the tree full with a height of 4.

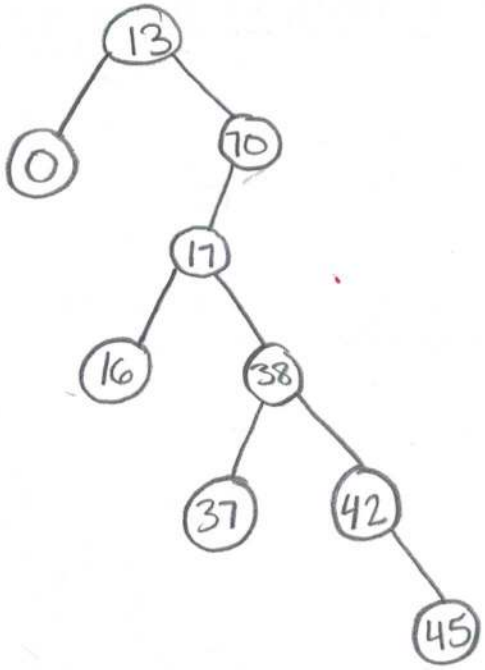
(e) Add (draw) nodes to the tree such that the tree is not perfect, but it is complete.

10 10 points

3 Tree Operations

6. (5 points) Draw the binary search tree that forms after inserting the following integers (13 inserted first):

13, 70, 17, 38, 16, 37, 42, 45, 0



5

5 points

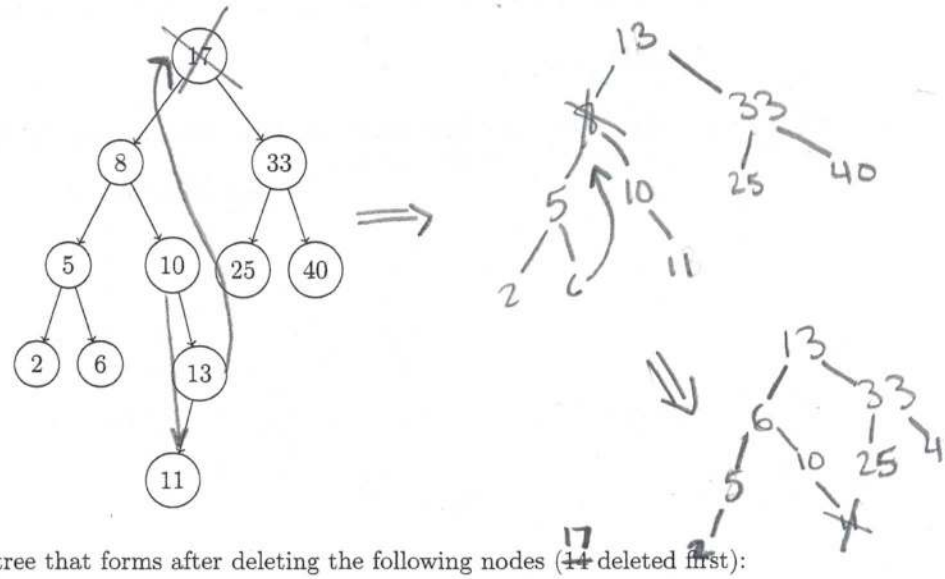
7. (5 points) Please give a post-order traversal of the tree you drew on the previous page.

post-order: T_L, T_R, root

0, 16, 37, 45, 42, 38, 17, 70, 13

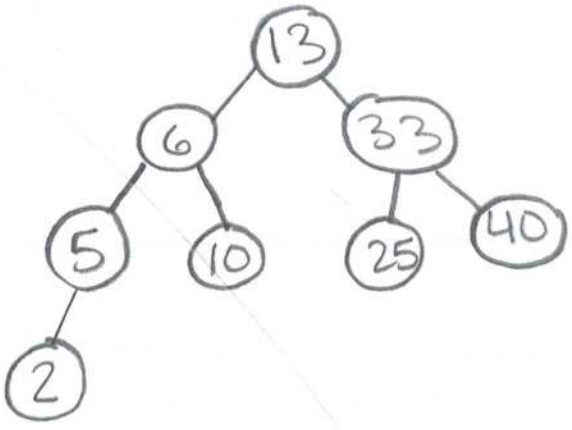
5

5 points



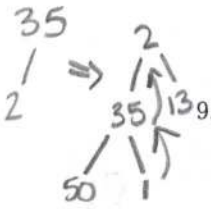
8. (7 points) Draw the binary search tree that forms after deleting the following nodes (~~14~~ deleted first):

17, 8, 11



7

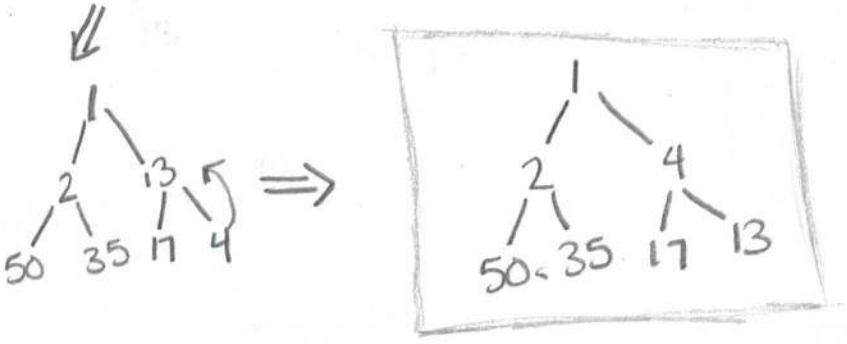
 7 points



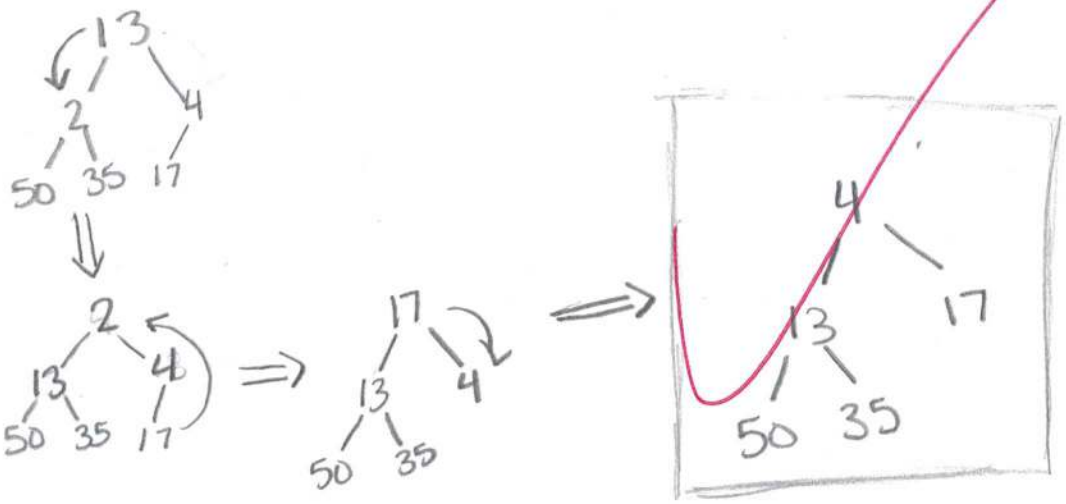
4 Heaps

9. (5 points) Draw the `Min-Heap` that forms after inserting the following integers (35 inserted first, and you can ignore the underlying array):

35, 2, 13, 50, 1, 17, 4



10. (5 points) Using the Heap you drew above, what would be the new root of the heap if you removed from the heap twice?



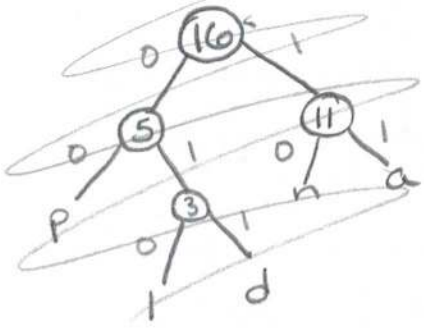
10 points

5 Huffman Coding

Consider a document that has the following symbols and frequencies:

Symbol	Frequency
d	2
a	7
n	4
l	1
p	2

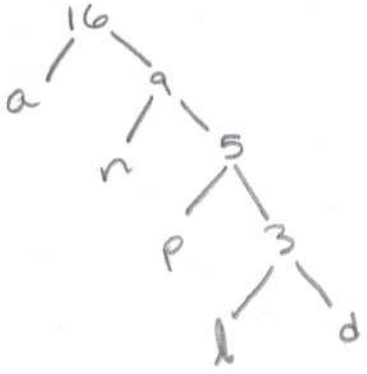
11. (7 points) Draw the Huffman Tree for these symbols. Break ties alphabetically.



12. (3 points) Encode the string "panda" into binary using this encoding.

p:00 a:11 n:10 d:011

panda:0011100111



3

6 Coding

On Febuary 26th, 2017, programmer Celestine Omin was detained by Customs and Border Protection. He was asked to prove he was a software engineer by writing a function to check if a binary tree was balanced.[1]

13. (12 points) Write a recursive method which, given the root of a tree or subtree, returns the height of the tree or subtree.

```
public static <E> int getHeight(Node<E> root) {
    if (root == null){
        return 0;
    }
    else if(root.left ==null && root.right != null){
        return 1+ getHeight(root.right);
    }
    else if (root.left !=null && root.right ==null){
        return 1+getHeight(root.left);
    }
    else if (root.left == null && root.right == null){
        return 1;
    }
    else{
        return MATH.max(getHeight(root.left)+1,
                        getHeight(root.right)+1);
    }
}
```

[1] It is unknown whether the interviewer was able to understand the answer. Celestine had not slept for 24 hours when he was detained.

12

14. (12 points) Write a method which, given the root of a tree, returns true if the tree is balanced and false if it is not. A tree is considered balanced if the height of its left subtree and right subtree differ by no more than 1 **and the subtrees are also balanced**. You may use the `getHeight` method from the previous question.

```
public static <E> boolean isBalanced(Node<E> root) {
    if (root == null) {
        return true;
    }
    else if (root.left==null && root.right ==null){
        return true;
    }
    else if (root.left !=null && root.right ==null){
        if (getHeight (root.left) >1 ){
            return false;
        }
        else {
            return true;
        }
    }
    else if (root.left ==null && root.right !=null){
        if (getHeight (root.right) > 1 ){
            return false;
        }else {
            return true;
        }
    }
    else {
        return isBalanced (root.right) && isBalanced (root.left);
    }
}
```

12 points

15. (11 points) Write a method which, given a `List` of strings, returns what Strings appeared in the array and how many times each word appeared. You may assume that each String holds a single word.

```
// INPUT   -> OUTPUT
// ["foo", "bar", "foo", "bar", "baz"] -> {"foo":2 , "bar":2, "baz":1}
// ["a", "b", "a","c", "a", "d"] -> {"a":3, "b":1, "c":1,"d":1}
public static Map<String,Integer> wordCount(List<String> words){
    Map<String, Integer> count=new HashMap<>();

    for(String word : words){
        if(!count.containsKey(word)){
            count.put(word, 1);
        }
        else{
            count.put(word, count.get(word)+1);
        }
    }
    return count;
}
```

11

7 Analysis

16. (5 points) We like to say that data structures built with hash tables can do $O(1)$ operations. Give a scenario where inserting or deleting an item from a hash table might take $O(n)$.

If an item is being inserted into a hash table but the max load of the table has been reached, the table will expand and rehash, which takes O(n).

17. (5 points) Some trees use nodes that have two data items, called x and y, where $x < y$, and three branches. The left branch stores all numbers $< x$, the right stores all number $> y$, and the middle all numbers between x and y.

How would having 3 branches instead of 2 affect the run time?

Hint: a binary search tree has $\log_2(n)$ runtime.

3 branches would slightly increase run time. For example, when adding a data item, three comparisons would need to be made with each root's children instead of 2 comparisons. The runtime would be $O(\log_3(n))$.

I think that's quicker

I'm just killing trees...

18. (1 point) Amuse me.

Two atoms walk into a bar.
One says to the other:
"I think I just lost an electron!"
"Are you sure?"
"I'm positive!"

 _____ 1 points