Topics for CSC 205 Quiz #4 **Wednesday, December 7th**

The following topics are especially important to review. Be sure you also understand the algorithm for writing the client program for the Job Queue program.

- What is a binary tree? What is a binary search tree?
- Tree terminology: root node, leaf node, left child, right child, parent, sibling, height, subtree, balanced, complete, full
- Calculating the maximum number of nodes in a balanced tree of a given height (Know the formula)
- Be able to draw a binary search tree for a given input set
- Be able to give the inorder, preorder, and postorder print traversals of a given binary search tree
- Be able to show what a binary search tree looks like after deleting a given node (Know how to handle all three scenarios discussed in class)
- Be able to show the array that could be used to store a binary search tree if the user is using the array based implementation of a binary search tree
- Writing a recursive method that takes the root as an input parameter and outputs the number of nodes in a tree or prints out a tree in ascending or descending order

Preorder - PLR
Inorder - LPR
Postorder - LRP

```
public void preorder(){
   preorder(root);
}
protected void preorder(TreeNode n
   if (node != null) {
      sysout (node.getItem());
      preorder(node.getLeft())
      preorder (node. get Right(
   }
}
```

Computer Science 205 Quiz #4

Wednesday, December 7th, 2016

50 points

32/50

Name : Yu-Ching

1. Consider the following batch file of jobs that are to be input into a queue-based time sharing system. Each line of input is made up of a job id, an arrival time, and a run time (in that order).

```
job1 01 03
job2 04 03
job3 07 04
```

(a) Complete the table below on the total time units for each of the five categories below. (6 points)

-3

Job Id	Arrival Time	Start Time	Run Time	Wait Time	Turnaround
job1	1		3		
job2	4		3		
job3	7		4		

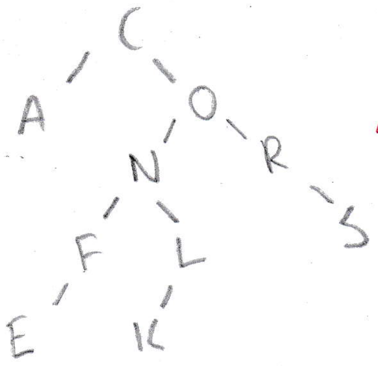
(b) Complete the summary table which calculates the average wait for this set of jobs, the cpu usage time, the cpu idle time, and the percentage of cpu usage. (4 points)

-4

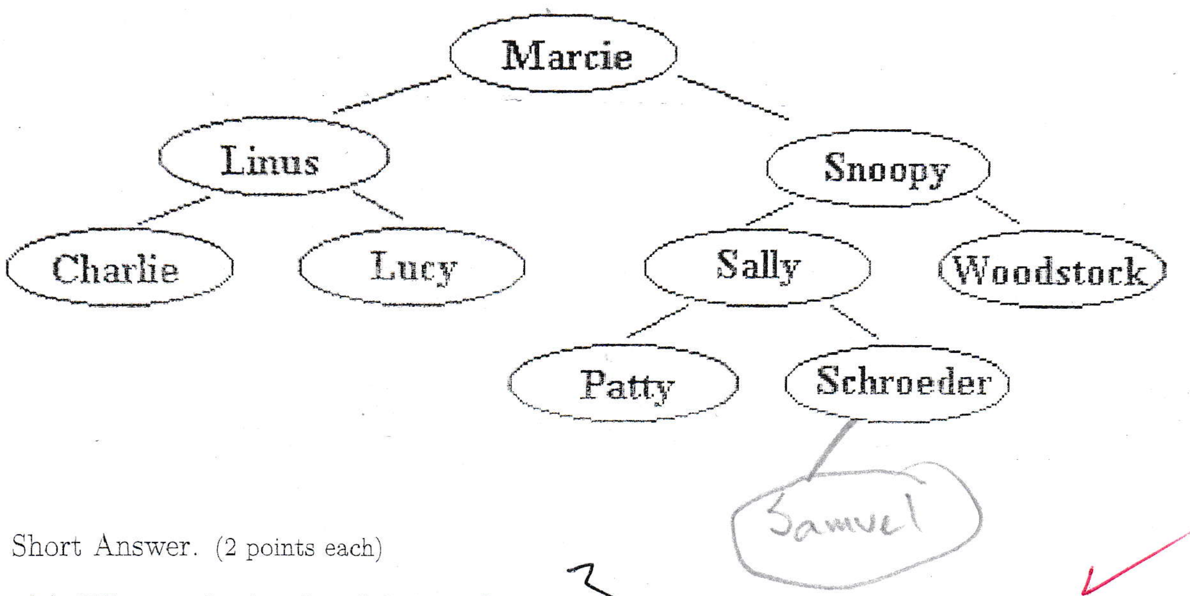
Avg. Wait	0
CPU Usage	~~0.16~~ 10
CPU Idle	0 . ~~44~~ 0
CPU Usage (%)	16 100

2. Sketch the binary search tree that would be created by the insertion of the following character components into a tree in the order listed below. (2 points)

C O R N F L A K E S



✓



3. Short Answer. (2 points each)

   (a) What is the *height* of this tree? 3 ✓

   (b) What is the average number of checks to find an item in this tree? 4 checks ✓

   size = 9 $\log_2 9 = 2^3 - 2^4$ 3 ~ 4 checks

~~4.~~ If the string `Samuel` is added to this tree, the tree will not be *balanced*. Explain why. Name all nodes that will now be out of balance. (2 points)

-2

The height on the left and right differs more than one.

Nods out of balance : ~~Schroeder, Woodstock~~

Woodstock, Snoopy, Lucy, Charlie, Sally, Linus, Marcie

5. Show the results of each of the following on the tree shown above. (2 points each)

   (a) postorder traversal LRP root at end

   Charlie, Lucy, Linus, Patty, Schroeder, Sally, Woodstock, Snoopy, Marcie ✓

   (b) inorder traversal LPR

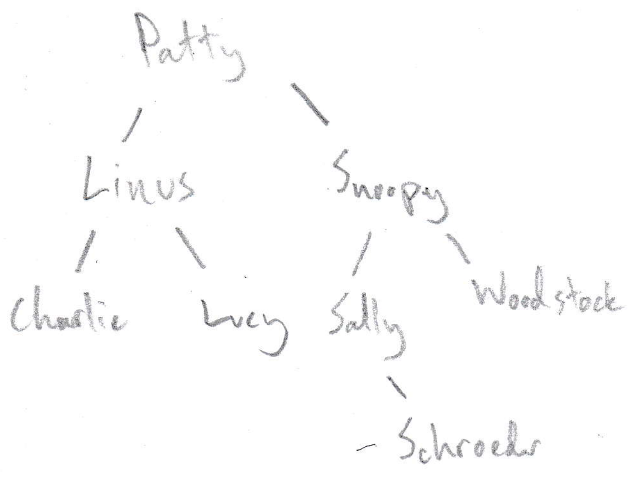
   Charlie, Linus, Lucy, Marcie, Patty, Sally, Schroeder, Snoopy, Woodstock ✓

   (c) preorder traversal PLR root at start

   Marcie, Charlie, Lucy, Linus, Patty, Schroeder, Sally, Woodstock, Snoopy ✓

largest on left, smallest on right

6. Show what this tree would look like if the root node were deleted. (2 points)



7. If you implemented this tree using an array-based implementation, fill in the contents of the array L for your tree below that could be used to store each node of this tree. (2 points)

Array L	Component	Lchild	Rchild
L[0]	Marcie	1	2
L[1]	Linus	3	4
L[2]	Snoopy	5	6
L[3]	Charlie	-1	-1
L[4]	Lucy	-1	-1
L[5]	Sally	7	8
L[6]	Woodstock	-1	-1
L[7]	Patty	-1	-1
L[8]	Schroeder	-1	-1

, size

8. Suppose you have a binary search tree with 1000 nodes. Answer each of the questions which follow. (2 points each)

(a) What is the smallest height this tree could be? Give the formula that would prove your answer to be true.

$2^{(h+1)} - 1 = \text{Size}$ $\log_2(h+1)$ $2^9 = 512$

$2^{(h+1)} = 1001$ $2^{10} = 1024$

Smallest height = 9 ✓

(b) How many null links does it have? How many edges (non-null links) does it have?

Every tree has (n+1) null links. 999

-1 Null links = 1001 ✓

(c) If this is a complete tree, how many more nodes on the last level would need to be added for the tree to become full?

-2

Number of leaves = $\frac{\text{height}+1}{2}$

$= \frac{999+1}{2}$

$= 500$

9. Given the BinarySearchTree class from lab. Answer the questions which follow.

(a) Suppose you were interested in adding a method duplicateCheck to check for two duplicate trees. Why would you set up both a public helper method and a private method to perform this task? That is, why even bother having a private version that is called by the public method? (4 points)

~~[illegible]~~

You want to have the public version to be called ~~anywhere~~ anywhere, but when to actually use it, the private method is there. The private method is there so it doesn't get called accidently

(b) Complete the body of the private recursive method duplicateCheck below. You will need to include the two base cases and the recursive case. You may assume a value-returning method named countNodes is available which returns an integer value representing the size of the tree pointed to by a node of type TreeNode. (6 points)

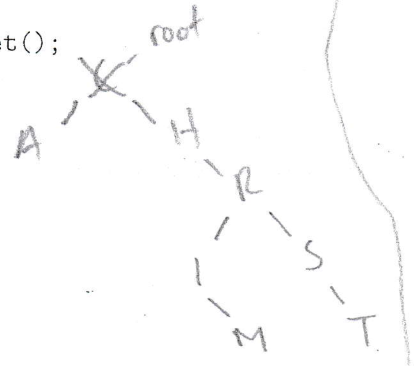
```
private boolean duplicateCheck(TreeNode t1, TreeNode t2)
{
        if (                                      )



        else if (                                     )



        else
        {
                KeyedItem key1 = (KeyedItem) t1.getItem();
                KeyedItem key2 = (KeyedItem) t2.getItem();
                String s1 = (String) key1.getKey();
                String s2 = (String) key2.getKey();









        }
}
```

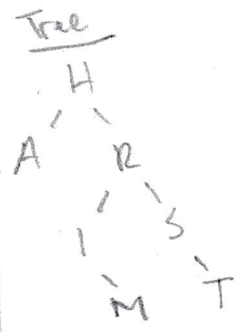
-6

10. Consider the code segment below which uses the `TreeSet` class from our class last Friday.

(a) Draw the tree that would be stored in RAM following this code segment with nine insertions and one deletions. (4 points)

```
TreeSet myTree = new TreeSet();
myTree.add ("C");
myTree.add ("H");
myTree.add ("R");
myTree.add ("I");
myTree.add ("S");
myTree.add ("T");
myTree.add ("M");
myTree.add ("A");
myTree.add ("S");
myTree.remove ("C");
```





(b) Suppose we sent off the root of the tree above to be printed by the special `mystery` traversal below. What would be the output? (2 points)

```
protected void mystery(TreeNode tNode)
{
        if (tNode != null)
        {
                mystery(tNode.getRight());
                System.out.println(tNode.getItem());
                mystery(tNode.getLeft());
        }
}
```

T, S, R, M, I, R, H, A