1. (4 points) What causes null pointer exceptions?

2. (4 points) Write a few lines of code that will cause a null pointer exception.

3. (4 points) Why is it important to keep your code simple?

4. (4 points) Since table sort is so fast, why don't we use it all the time?

5. (4 points) Why must you first count the neighbors of all the Cells and then update them (instead of counting and updating each in order)?

6. (10 points) Write pseudocode for bubble sort. What is its running time, in θ notation?

7. (15 points) Write pseudocode to evaluate an arbitrary postfix expression. Assume you are provided with a `Stack`, an emitter (that emits whatever you pass it) and a tokenizer (with `nextToken()` and `hasMoreTokens()`).

8. (5 points) What are the two steps in a proof by induction?

   i

   ii

9. (10 points) What is the definition of a binary tree?

10. (20 points) Fill in insertRoot (which should change an empty tree to one with the passed value at the root), and write a method, called size, which returns the number of non-empty subtrees (i.e. the number of values in a tree) -- make sure you won't get a null-pointer exception when you use right and left afterwards!

```
public class BinaryTree {
    private Integer root=null;
    private BinaryTree right=null;
    private BinaryTree left=null;

    public BinaryTree() {
    }

    boolean isEmpty() {
        return root == null;
    }

    public void setRoot(Integer n) {









    }
```

11. (10 points) Use an `ArrayList<Object>` to write a complete Queue class with methods `void enqueue(Object)`, `Object dequeue()` and `boolean isEmpty()`.

12. (10 points) Assuming you have a `Queue` class, implement Stack (push, pop, and isEmpty()) utilizing not more than 2 `Queues` (no other data structures allowed!).

13. (10 xtra credit) Do the previous problem recursively using only one Queue (on back of page 3).