1. (4 points) Write some code, which, when executed, will generate a NullPointerException.

2. (4 points) What are the two steps in a proof by induction?
   i
   ii

3. (5 points) When is an algorithm, A, with running time on input of length n, $T_A(n)$, said to be $\Theta(f(n))$?

4. (5 points) What is the running time of this pseudocode?

   ```java
   for each element, i, of the list
       for each element, j, of the list
           if (i != j){
               for each element of the list that is smaller than the ith {
                   increment its count
               } // counting for
           } //if
       } // inner for
 } // outer for
```

5. (4 points) You have two sorts which you analyze to be $\Theta(n)$; you time them and discover that one runs a billion times faster than the other. Could they actually both be $\Theta(n)$? Explain.

6. (10 points) Use an `ArrayList<Object>` to write a complete `Stack` class with methods `void push(Object)`, `Object pop()` and `boolean isEmpty()`.
7. (5 points) Declare a class, named MyList, which contains an array, named list, with a constructor, which is passed the length of the list, allocates the storage, and initializes each element as follows -- the first should be set to 1, the second, to 2, ..., and the nth, to n.

8. (10 points) Write a class, XList (X for eXpandable), based on ArrayList, with an XList(MyList) (see previous question) constructor, that adds all the values from the MyList to itself.

9. (5 points) (continuing…) Write a sum() method for XList, which returns the sum of all its elements.

10. (3 points) (...still continuing…) If this code is executed, what will it print?

```java
XList list = new XList(new MyList(2000));
sout("The sum is: " + list.sum());
```
11. (10 points) Write a BST (binary search tree) class, including a default constructor and one that sets the root, plus, boolean isEmpty() and void insert(int).

12. (10 points) Assume you have working BST (see previous), and XList (see 8) classes; implement void treeSort(XList). Note: you may write the method to traverse the tree here or in BST, just indicate where it goes.

13. (5 points) Write a complete Board class for tic tac toe which represents the board as a 2D array of ints. Write just the default constructor which sets the initial state to
14. (15 points) Write an NaryTree class, which has a root Board, and an NaryTreeList of children. Include NaryTree(Board), that constructs the tree with that Board as the root; assume Board has boolean gameOver(), and BoardList generateNextLegalBoards().

15. (5 points) Assume Board has an int getScore() method. Write NaryTree findMax(), which searches the entire tree and returns the NaryTree whose root has the highest score.

16. (10 points extra credit!!!!!!) Write void enqueue() for a Queue using only one Stack. Assume int dequeue {return pop();} But, you must make it fit on this page somewhere!