[**Closed book and notes.**] Show all of your work clearly in the space provided. Be sure to **read each problem carefully**. Note that the exam is double sided. **Points given for short answer questions will be scored much like answers to interview questions would be evaluated. An answer that makes you more likely to be hired will receive a higher score.**

**1.** (15 points) Explain the criteria for binary trees that are full, perfect, and complete.

**2.** (8 points) Describe the design flaw found in the java.util.Stack class.

**3.** (7 points) When implementing a recursive method, what is a base case? Why is it needed?

**4.** (5 points) What data structure is FIFO associated with? What does FIFO stand for?

**5. (a)** (15 points) Recall that our BinarySearchTree class had an inner, Node, class that contained three attributes: value, lKid, and rKid. Implement the recursive version of BinarySearchTree.countLeaves() that is called by the method below such that the method returns the number of leaves in the tree.

```
public int countLeaves() {
    return countLeaves(root);
}
```

**(b)** (5 points) Use big-oh notation to describe the overall worst case time complexity for your algorithm. Be sure to explain your reasoning.

**6. (a)** (15 points) Implement the recursive version of BinarySearchTree.toString() that is called by the method below such that the following method returns a string containing all of the elements in the binary search tree listed in order (separated by commas).

```
public String toString() {
    String result = toString(root);
    result = result.length()<3 ? "" : result.substring(0, result.length()-2);
    return "[" + result + "]";
}
```

**(b)** (5 points) Use big-oh notation to describe the overall worst case time complexity for your algorithm. Be sure to explain your reasoning.

**7.** A palindrome is a phrase that reads the same forward and backward. For example:

- radar
- Straw? No, too stupid a fad. I put soot on warts.

**(a)** (20 points) Notice that punctuation, capitalization, and spacing are ignored when determining whether a phrase is a palindrome or not. Use either or both the Stack<E> and Queue<E> classes developed in lecture to implement the following method. Hint: Character.isLetter('A'); returns `true`, and Character.isLetter('!'); returns `false`.

```
// Return true if and only if the phrase is a palindrome
public static boolean isPalindrome(String phrase) {
```

Part **(a)** cont. . .

**(b)** (5 points) Use big-oh notation to describe the overall worst case time complexity for your algorithm. Be sure to explain your reasoning.