# SEVENTH QUIZ

You have 15 minutes from the start of class to complete this quiz. Give partial answers if you can't give complete ones. Read the questions with care; work with deliberate speed. Don't give us more than we ask for. The usual instructions apply. Good luck!

**Problem 1** (6 points)

Here is a recursive method:

```
boolean inTree (int item, BST tree)
{
     if (tree.isEmpty()) return false;
     else if (item == tree.getRoot()) return true;
     else if (item < tree.getRoot()) return inTree(item, tree.getLeft());
     else /* item > root          */ return inTree(item, tree.getRight());
}
```

**(a)** (4 points) Give the recurrence relation that describes the amount of time this function takes to finish, counting comparisons (== or < or >). Assume that a BST is a well-balanced binary search tree of *n* ints; also assume the "worst case" that the item is never found (i.e., that you always have to look all the way to the bottom of the tree).

$T_0 =$

$T_n =$

**(b)** (2 points) What is the O-notation of this code in terms of its argument *n*? You don't have to solve the recurrence; you can just give the O-notation from your understanding of how the algorithm works.

**Problem 2** (2 points)

List two ways in which the end-users should be involved (*before* they start using the completed system) in a user-centered design of a human-computer interface.

**Problem 3** (2 points)

If you only have a single processor, what's the point of writing a program using concurrency (Threads and Runnables in Java)? In other words, why pretend you (might) have more processors than you actually do? Answer in one or two brief English sentences.

**Problem 4** (10 points)

In our amusement park simulator, a customer could come to the park with an entrance/arrival time that's after the park closes. Just as in real life, that customer wouldn't get into the park (on that day, at least).

Attached is a copy of the original AmusementPark class. Modify the code (by writing new code at the appropriate point on the copy) so that when the simulation is completed, it prints the number of customers who are still waiting to enter the park. (Of course that number might be zero.) This can be done in one line of code.

Your added output might look like this:

```
35 customers were still waiting when the park closed.
```

(You should not need to see or modify any other classes that make up the simulator.)