Final Exam Practice Problems

These problems are not intended to reflect the difficulty nor length of the final exam. They are simply problems that test various topics that have a good chance of appearing on the final exam.

1. Write a function that takes an array (or ArrayList) of integers and determines if the array has at least one duplicate element (some integer that appears more than once). For instance, an array like {4, 7, 5, 1} has no duplicate elements, while the array {4, 1, -5, 1, 6} does have a duplicate. Any integer may appear anywhere in the array.

   The declaration line for the function is:

   ```
   public static boolean hasDuplicate(int[] array)
   ```

2. What is the big-oh time of your function above?

3. Rewrite the function from problem 1, now assuming that you know ahead of time that the array is **sorted in ascending order**. Hint: while the code you wrote for #1 will also work here, you should be able to write a more efficient algorithm.

4. What is the big-oh time of your function above?

5. Define a **recursive** function that, given any positive integer n, returns the sum of all the odd integers between 1 and n (inducing n itself, if n is odd). For instance, this function should return 9 if n=5, because 1 + 3 + 5 = 9. You may not use a formula to solve this problem in one step, although there is a simple formula. Do not use loops. You must use recursion.

   The declaration line for the function is `public static int sumOdd(int n)`

6. Suppose I have the following recursive function:

   ```
   public static void silly(int n) {
      if (n >= 9) {
         System.out.println(n);
      }
      else {
         System.out.println(n);
         silly(n+1);
         System.out.println(n);
         silly(n+2);
      }
   }
   ```

   Show the output that is printed when we call `silly(7).`

7. Write a recursive function, `public static String stars(int n)` that returns a string consisting of n asterisks in a row. For example, stars(5) should return the string "*****". Do not use loops.

8. Assume we have a Rectangle class with four instance variables for the coordinates of the top left corner (int topLeftX, int topLeftY), and the width and height of the rectangle (int width, int height).

   Write a method of this Rectangle class that determines whether "this" rectangle is completely contained within another Rectangle (that is, if the two rectangles were drawn on the screen, the first would be completely inside the boundary of the second)

   ```
   public class Rectangle {
     private int topLeftX, topLeftY, width, height;

     public boolean isInside(Rectangle otherRect) { }  // write this method
   }
   ```

9. Design a Time class to represent a time that might be shown on a watch or clock. A Time needs to represent an hour, a minute, and whether it is AM or PM. Design a class to store this information. You can pick the instance variables. Your class should have the following methods:

   * public getters for the hour, minute, and AM or PM.
   * a toString() method that returns a nicely-formatted String representing the time, like "3:20 PM" or "10:04 AM"
   * a method isEarlier(Time otherTime) that returns a boolean that is true when "this" time is earlier than the otherTime argument.

10. You have an algorithm that processes an array. The algorithm takes **roughly** the following amounts of time to run:

    * Array of 5,000 elements: 5 seconds
    * Array of 10,000 elements: 22 seconds
    * Array of 20,000 elements: 90 seconds
    * Array of 80,000 elements: 1451 seconds

    Roughly how long would it take, **in hours**, for your algorithm to process an array of 150,000 items?

    What do you think the big-oh time of this algorithm is?

11. Write a function that takes a 2-d array of integers and determines if there is a number in the array such that its four neighbors above, below, left, and right are all larger than it. For instance, the left matrix below has such a number but the right matrix does not. (Ignore any numbers in the array that do not have all four neighbors.)

```
6 7 9 6          6 7 9 6
3 4 2 3          3 4 5 3
8 6 7 2          8 6 7 2
```

declaration line: `public static boolean findNumberWithLargerNeighbors(int[][] array)`