Mehran Sahami
CS106A

Handout #49
December 4, 2017

Additional Practice Problems

These additional problems are courtesy of Keith Schwarz

Problem 1: Strings (15 Points)

An *isogram* is a word that contains no repeated letters. For example, the word “computer” is an isogram because each letter in the word appears exactly once, but the word “banana” is not because 'a' and 'n' appear three times each. “Isogram” is itself an isogram, but “isograms” is not because there are two copies of 's'.

There are many long isograms in English; for example, “uncopyrightable” and “computerizably.” Your job is to write a method that, given a list of all the words in the English language, finds out what the longest isogram actually is. Write a method

```
private String longestIsogram(ArrayList<String> allWords)
```

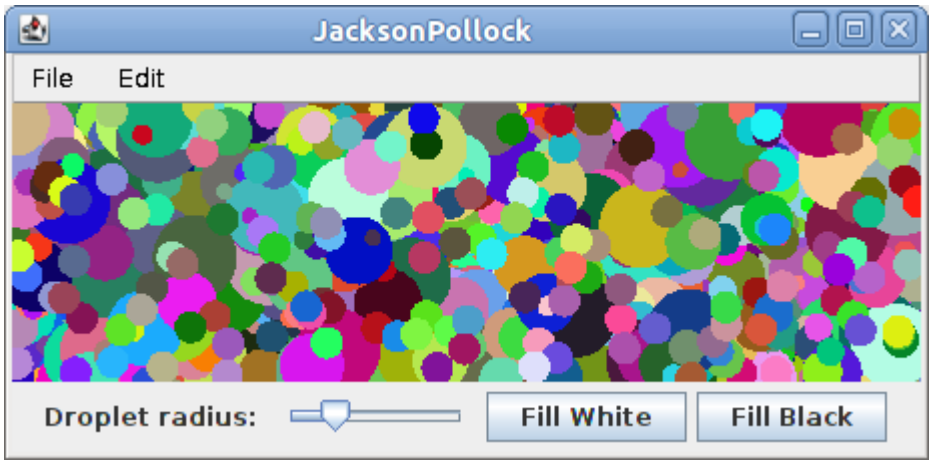
that accepts as input a `ArrayList<String>` containing all words in English (stored in lowercase) and returns the longest isogram in the list. If multiple words are tied as the longest isogram, feel free to return any one of them.

```
private String longestIsogram(ArrayList<String> allWords) {
```

Problem 2: Graphics and Interactivity (25 Points)

In this problem, you'll build a program that draws artwork in the style of the abstract expressionist painter Jackson Pollock. Pollock created paintings by laying the canvas down on the floor of his studio, then throwing paint of different colors onto it. The resulting paintings contain a mishmash of colors that are artistically and aesthetically interesting.

Your task is to write a program that simulates randomly-thrown droplets of colored paint landing on a canvas. Below is a screenshot of this program:



As soon as the program starts up, it begins drawing randomly-positioned circles on the canvas, each of which represents a drop of paint. The center of each circle is chosen as a random point inside the canvas, so the entire circle won't necessarily fit inside the window. In order to watch the art evolve over time, you should pause for **`PAUSE_TIME`** milliseconds after each drop of paint. Each circle's color should be chosen at random.

The radius of each circle should be determined by the value of a **`JSlider`** at the bottom of the window. The slider should range between the values **`MIN_RADIUS`** and **`MAX_RADIUS`**, and its default value should be **`DEFAULT_RADIUS`**. This slider should have a label to its left that reads "**Droplet radius:**" so that users understand what it controls.

If the user clicks the **Fill White** button, then the display should be filled with a solid white color, representing what would happen if you covered the canvas in a complete coat of white paint. The **Fill Black** button is similar, except that it will fill the canvas with black paint.

Write your implementation on the next page, and feel free to tear out this sheet as a reference.

```
import /* ...lots of things you don't need to worry about... */

public class JacksonPollock extends GraphicsProgram {
    /** Amount of time to pause between droplets, in milliseconds. */
    private static final double PAUSE_TIME = 1.0;

    /** Minimum, maximum, and default radius of each drop of paint. */
    private static final int MIN_RADIUS = 3;
    private static final int MAX_RADIUS = 20;
    private static final int DEFAULT_RADIUS = 7;
```

Problem 3: Random Numbers (25 Points)

Suppose you want to hold a never-ending birthday party, where every day of the year someone at the party has a birthday. How many people do you need to get together to have such a party?

Your task in this program is to write a program that simulates building a group of people one person at a time. Each person is presumed to have a birthday that is randomly chosen from all possible birthdays. Once it becomes the case that each day of the year, someone in your group has a birthday, your program should print out how many people are in the group, then should exit.

In writing your solution, you should assume the following:

- There are 366 possible birthdays (this includes February 29).
- All birthdays are equally likely, including February 29.

You might find it useful to represent birthdays as integers between 0 and 365, inclusive.

```
import acm.program.*;

public class NeverendingBirthdayParty extends ConsoleProgram {
```

Problem 4: Arrays (25 points)

A *magic square* is an $n \times n$ grid of numbers with the following properties:

1. Each of the numbers 1, 2, 3, …, n^2 appears exactly once, and
2. The sum of each row and column is the same.

For example, here is a 3×3 magic square, which uses the numbers between 1 and $3^2 = 9$:

4	9	2
3	5	7
8	1	6

and here is a 5×5 magic square, which uses the numbers between 1 and $5^2 = 25$:

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

Write a method

```
private boolean isMagicSquare(int[][] square, int n);
```

that accepts as input a two-dimensional array of integers (which you can assume is of size $n \times n$) and returns whether or not it is a magic square.