Eric Roberts Handout #2
CS 298 April 26, 2017

Assignment #1

Due: Wednesday, May 10, in class

The topic for today's class is the JavaPPTX package, which makes it possible to create PowerPoint slides from Java. This assignment asks you to write two animations, one that uses the standard capabilities of the **`edu.stanford.cs.pptx`** package and one that uses the code-tracing facilities in **`edu.stanford.cs.pptx.code`**. If you have your own ideas of interesting animations, you can substitute your own animation for either of these as long as you talk with me first.

Problem 1. Animate Euclid's algorithm

Euclid's algorithm for calculating the greatest common divisor of two integers is much easier to understand when you can watch the steps involved. The iterative version of the process is described in the following excerpt from my book *Programming Abstractions in C*++, which is used as the textbook in CS 106B:

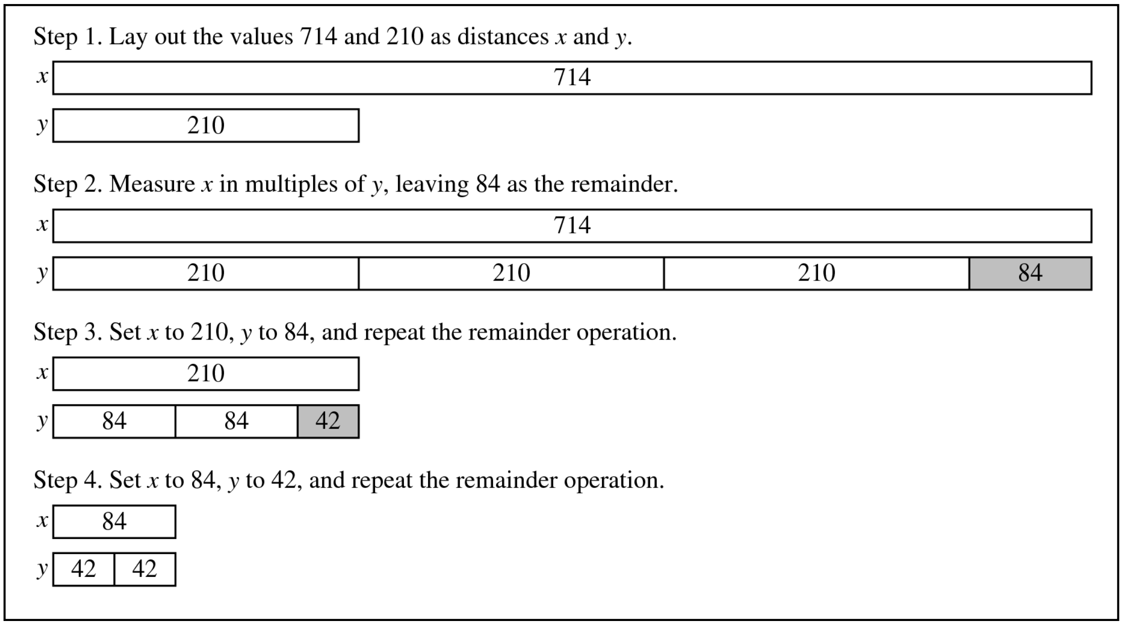
> One of the earliest known algorithms worthy of that distinction is named for the Greek mathematician Euclid, who lived in Alexandria during the reign of Ptolemy I (323–283 BCE). In his great mathematical treatise called *Elements,* Euclid outlines a procedure for finding the ***greatest common divisor*** (or ***gcd*** for short) of two integers *x* and *y,* which is defined to be the largest integer that divides evenly into both. For example, the gcd of 49 and 35 is 7, the gcd of 6 and 18 is 6, and the gcd of 32 and 33 is 1. In modern English, Euclid's algorithm can be described as follows:
>
> 1. Divide *x* by *y* and compute the remainder; call that remainder *r*.
> 2. If *r* is zero, the algorithm is complete, and the answer is *y*.
> 3. If *r* is not zero, set *x* to the old value of *y*, set *y* equal to *r*, and repeat the process.

Figure 1 at the top of the next page illustrates the steps involved in calculating the greatest common divisor of 714 and 210.

Your task in this assignment is to write a program **`PPEuclid.java`** that creates a single PowerPoint slide that displays the rectangles in Figure 1. The program should animate those rectangles so that the following actions occur:

1. The slide appears with the rectangles for the original values for *x* and *y,* scaled so that *x* takes up the full width after leaving a suitable margin on each side.
2. Create new rectangles that mark off the value of *x* in units of *y,* leaving some remainder *r*. The first rectangle should occur on a mouse click; the remaining rectangles should appear one after another using a 0.2 second delay.
3. If the remainder is not 0, your animation should show two new rectangles lower on the slide with lengths *y* and *r,* and then go back to step 2.

Figure 1. Geometric interpretation of Euclid's algorithm for 714 and 210



Step 1. Lay out the values 714 and 210 as distances *x* and *y*.

Step 2. Measure *x* in multiples of *y*, leaving 84 as the remainder.

Step 3. Set *x* to 210, *y* to 84, and repeat the remainder operation.

Step 4. Set *x* to 84, *y* to 42, and repeat the remainder operation.

Feel free to leave all the text out of the slide except for the numbers inside the rectangles. These numbers are easy to add simply by calling **`setText`** on the **`PPRect`** object. The other labels are easier to add later using PowerPoint's standard tools.

Problem 2. Animate the recursive formulation of Euclid's algorithm

Use the **`PPFactTrace.java`** program from the examples directory as a model, write a program **`PPEuclidTrace.java`** that traces all calls and stack frames in the recursive implementation of **`gcd`**, shown here in its JavaScript form:

```
function gcd(x, y) {
   if (y === 0) {
      return x;
   } else {
      return gcd(y, x % y);
   }
}
```

Your animation should also include a stack frame for the following test function:

```
function TestEuclidGCD() {
   console.log("gcd(714, 210) = " + gcd(714, 210));
}
```