Alisha Adam & Rohit Talreja
CS 106A – Summer 2016

Solutions for Section #2 – Console & Graphics Programs

Based on handouts by Marty Stepp and Keith Schwarz

1. Fibonacci Sequence

```
import acm.program.*;

public class Fibonacci extends ConsoleProgram {

	/* Defines the largest term to be displayed */
	private static final int MAX_TERM_VALUE = 10000;

	public void run() {
		println("This program lists the Fibonacci sequence.");
		int t1 = 0;
		int t2 = 1;
		while (t1 <= MAX_TERM_VALUE) {
			print(t1 + " ");
			int t3 = t1 + t2;
			t1 = t2;
			t2 = t3;
		}
	}
}
```

2. Fizz Bazz Buzz

```
import acm.program.*;

public class FizzBazzBuzz extends ConsoleProgram {
	public void run() {
		int numRounds = readInt("How many rounds? ");
		for (int i = 0; i < numRounds; i++) {
			/* Check for divisibility by 15. If we don't do this test
			 * first, then we might print out the wrong message.
			 */
			if (i % 15 == 0) {
				println("Buzz");
			}
			/* Now, check for all the remaining conditions. */
			else if (i % 3 == 0) {
				println("Fizz");
			} else if (i % 5 == 0) {
				println("Bazz");
			} else {
				println(i);
			}
		}
	}
}
```

3. Mystery function trace

Call	Output
mystery(8);	8 6
mystery(-3);	-3 4
mystery(1);	1 3
mystery(0);	0 0

4. Optical Illusion

```
import acm.program.*;
import acm.graphics.*;

public class PhantomBoxes extends GraphicsProgram {

	/* The number of boxes in each row or column. */
	private static final int BOXES_PER_SIDE = 4;

	/* The width and height of each box. */
	private static final double BOX_SIZE = 50;

	/* The horizontal and vertical whitespace between boxes. */
	private static final double BOX_SPACING = 8;

	public void run() {
		/* Compute the total width/height of one row of boxes. This includes
		 * BOXES_PER_SIDE boxes of width BOX_SIZE, plus the number of
		 * spaces in-between the boxes times the box spacing.
		 */
		double sideSize = BOXES_PER_SIDE * BOX_SIZE +
			(BOXES_PER_SIDE - 1) * BOX_SPACING;

		/* Determine the center of the screen. */
		double centerX = getWidth() / 2.0;
		double centerY = getHeight() / 2.0;

		/* Determine x, y coordinates of the upper-left corner of the grid */
		double xStart = centerX - sideSize / 2.0;
		double yStart = centerY - sideSize / 2.0;

		/* Draw the grid of boxes. */
		for (int row = 0; row < BOXES_PER_SIDE; row++) {
			for (int col = 0; col < BOXES_PER_SIDE; col++) {
				/* Determine the position of this box, starting at the
				 * upper-left corner of the upper-left box and adding
				 * extra space to skip all intervening boxes.
				 */
				double x = xStart + col * (BOX_SIZE + BOX_SPACING);
				double y = yStart + row * (BOX_SIZE + BOX_SPACING);

				drawBox(x, y);

			}
		}
	}

	/* Add a filled box. */
	private void drawBox(double x, double y) {
		GRect box = new GRect(x, y, BOX_SIZE, BOX_SIZE);
		box.setFilled(true);
		add(box);
	}
}
```