Jerry Cain
CS 106AX

Handout #4S
October 6, 2023

Solution to Section #1

Portions of this handout by Eric Roberts, Patrick Young, and Ryan Eberhardt.

Solution 1: Narcissistic Numbers

```
/**
 * Function: Narcissistic
 * ---------------------
 * Returns true if and only if the supplied number is
 * narcissistic, as outlined in the statement of Problem 1.
 */
function Narcissistic(number) {
   let len = countDigits(number);
   let original = number;
   let sum = 0;
   while (number > 0) {
      let digit = number % 10;
      sum += Math.pow(digit, len);
      number = Math.floor(number/10);
   }
   return sum === original;
}

/**
 * Function: countDigits
 * --------------------
 * Returns the number of digits in the supplied number.
 */
function countDigits(number) {
   let count = 0;
   while (number > 0) {
      count++;
      number = Math.floor(number/10);
   }
   return count;
}
```

Solution 2: Random circles

```
/*
 * File: RandomCircles.js
 * ----------------------
 * This program draws a set of 10 circles with different sizes,
 * positions, and colors.  Each circle has a randomly chosen
 * color, a randomly chosen radius between 5 and 50 pixels,
 * and a randomly chosen position on the canvas, subject to
 * the condition that the entire circle must fit inside the
 * canvas without extending past the edge.
 */

/* Constants */

const NCIRCLES = 10;
const MIN_DIAMETER = 10;
const MAX_DIAMETER = 100;
const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 300;

/* Main function */
function main() {
    let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
    for (let i = 0; i < NCIRCLES; i++) {
       let d = randomInteger(MIN_DIAMETER, MAX_DIAMETER);
       let circle = makeRandomColoredCircle(d);
       let x = randomInteger(0, gw.getWidth() - d);
       let y = randomInteger(0, gw.getHeight() - d);
       gw.add(circle, x, y);
    }

    handleInteractiveDrawing(gw);
}

/* Returns the distance between two given points. */
function getEuclidianDistance(x0, x1, y0, y1) {
    return Math.sqrt(Math.pow(x1 - x0, 2) + Math.pow(y1 - y0, 2));
}

/* Returns a GOval with a random color and the provided
 * diameter. */
function makeRandomColoredCircle(diameter) {
    let circle = new GOval(diameter, diameter);
    circle.setFilled(true);
    circle.setColor(randomColor());
    return circle;
}
```

```
/* Installs mousedown and drag event handlers in order to create a
 * circle on mousedown, and increase the size on drag. */
function handleInteractiveDrawing(gw) {
    let circle = null;
    let centerX = null;
    let centerY = null;

    let mouseDownHandler = function(e) {
        circle = makeRandomColoredCircle(0);
        centerX = e.getX();
        centerY = e.getY();
        gw.add(circle, centerX, centerY);
    };
    gw.addEventListener("mousedown",  mouseDownHandler);


    let dragHandler = function(e) {
        let radius = getEuclidianDistance(centerX, e.getX(),
             centerY, e.getY());
        circle.setLocation(centerX - radius, centerY - radius);
        circle.setSize(radius * 2, radius * 2);
    };
    gw.addEventListener("drag",  dragHandler);
}
```

Note: on some runs of the program, you might not *see* 10 circles because some circles will be drawn white or be drawn on top of previously drawn one, potentially blocking them entirely from view.

Solution 3: Tracing function execution

The output of `CalculateBill.js` is:

```
Your total before tip is: $100.
Your final price is: $106.
```

For this problem, there are several concepts that we should understand.

(1) Parameters
Look at the following two lines:

```
let finalPrice = calculateBill(numSalads, numPizzas);
function calculateBill(numPizzas, numSalads) { ... }
```

Here, note that we actually swap `numPizzas` and `numSalads` when passing them as parameters, so inside `calculateBill`, `numPizzas` should be 4 and `numSalads` should be 6.

There is no connection between the names given to variables in one function, and the names of the parameters which those variables are assigned to during a function call. **When passing a variable as an argument to a function, JavaScript assigns the first argument to the first parameter, the second argument to the second parameter, and so forth, regardless of the names they are given.** Giving confusing names to parameters though, while certainly possible, constitutes bad style and should be avoided.

(2) Closure
Remembering that when you define an inner function within an existing function, we call that a closure, and **closures can access the variables of the outer function**. This means:

- In `calculateBill`, `addSaladCosts` and `addPizzaCosts` can both access and modify `total` without passing `total` in as a parameter.
- However, `addTax` and `addTip` both require passing `total` in as a parameter because they are defined outside of `calculateBill`.
- Finally, note that even though `total` is changed in `addTax`, it will not affect the value of `total` in `calculateBill`.