Jerry Cain
CS 106AJ

Handout #14S
October 10-11, 2018

Solutions for Section #2

Solution 1: Perfect Numbers

```
/*
 * File: PerfectNumber.js
 * ----------------------
 * This program displays the perfect numbers in a given range.
 */

/* Constants */
const MAX_NUMBER = 9999;

/* Prints out the perfect numbers in the range [1, MAX_NUMBER] */
function PerfectNumbers() {
   console.log("Perfect Numbers between 1 and " + MAX_NUMBER + ":");
   for (let n = 1; n <= MAX_NUMBER; n++) {
      if (isPerfect(n)) {
         console.log(n);
      }
   }
}

/*
 * Function: isPerfect
 * -------------------
 * Returns true if the integer n is perfect, which is to say that it is
 * equal to the sum of its proper divisors, and false otherwise.
 */
function isPerfect(n) {
   let sum = 0;
   for (let i = 1; i < n; i++) {
      if (n % i === 0) {
         sum += i;
      }
   }
   return (sum === n);
}
```

Solution 2: Finding Easter

```
/*
 * File: Easter.js
 * ------------------
 * This program uses Gauss' algorithm to compute the date of Easter for
 * years between 1700 and 1899.
 */

/* Constants */
const YEAR = 1800;

/* Prints out the date of Easter in year YEAR. */
function Easter() {
   if (YEAR < 1700 || YEAR > 1899) {
      console.log("Whoops! Gauss' algorithm doesn't work for year "
         + YEAR + ".\nPlease choose a year between 1700 and 1899.");
   } else {
      console.log("In " + YEAR + ", Easter fell on " +
                  findEaster(YEAR) + ".");
   }
}

/*
 * Function: findEaster
 * ------------------
 * Returns a string showing the date of Easter in the specified year.
 * For example, calling findEaster(1800) returns the string "April 13"
 * because that is the date of Easter in the year that Gauss published
 * his algorithm.
 */
function findEaster(year) {
   // Step I:
   let a = year % 19;
   let b = year % 4;
   let c = year % 7;

   // Step II:
   let d = (19 * a + 23) % 30;

   // Step III:
   let eDividend = (2 * b) + (4 * c) + (6 * d) + 3;
   if (year >= 1800) {
      eDividend++;
   }
   let e = eDividend % 7;

   // Determine date of Easter:
   if (d + e > 9) {
      return "April " + (d + e - 9);
   } else {
      return "March " + (22 + d + e);
   }
}
```

Solution 3: Drawing a face

```
/*
 * File: RobotFace.js
 * --------------------
 * This program draws a robot face using GRects and GOvals.
 * We make sure to define constants at the top of our program instead
 * of using magic numbers. We also write the program in terms of
 * reusable and general methods drawRectangle and drawCircle.
 */

/* Constants for the drawing */
const HEAD_WIDTH = 150;
const HEAD_HEIGHT = 250;
const EYE_RADIUS = 10;
const MOUTH_WIDTH = 60;
const MOUTH_HEIGHT = 20;
const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 200;

/* Draw the entire face centered in the graphics window */
function main(){
   let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
   let cx = gw.getWidth()/2;
   let cy = gw.getHeight()/2;
   addHead(cx - HEAD_WIDTH/2, cy - HEAD_HEIGHT/2, gw);
   addEye(cx - HEAD_WIDTH/4, cy - HEAD_HEIGHT/4, gw);
   addEye(cx + HEAD_WIDTH/4, cy - HEAD_HEIGHT/4, gw);
   addMouth(cx - MOUTH_WIDTH/2, cy + HEAD_HEIGHT/4, gw);
}

/*
 * Add a head with top left at position x,y. Adding a head consists
 * of drawing a rectangle with the given width, height, and color.
 */
function addHead(x, y, gw){
   drawRectangle(x, y, HEAD_WIDTH, HEAD_HEIGHT, "Grey", gw);
}

/*
 * Add an eye centered at cx, cy. Adding an eye consists of drawing
 * a circle with the given radius and color.
 */
function addEye(cx, cy, gw){
   drawCircle(cx, cy, EYE_RADIUS, "Yellow", gw);
}

/*
 * Add a mouth with top left at x,y. Adding a mouth consists of
 * drawing a rectangle with given width, height and color.
 */
function addMouth(x, y, gw){
   drawRectangle(x,y, MOUTH_WIDTH, MOUTH_HEIGHT, "White", gw);
}
```

```
/*
 * This function draws a general rectangle with its top left
 * at position x,y with a specified width, height and color.
 */
function drawRectangle(x, y, width, height, color, gw){
   let rect = GRect(x, y, width, height);
   rect.setFilled(true);
   rect.setColor(color);
   gw.add(rect);
}

/*
 * This method draws a general circle centered at (cx,cy),
 * with a given radius r and a Color c.
 */
function drawCircle(cx, cy, r, color, gw){
   let x = cx - r;
   let y = cy - r;
   let circle = GOval(2 * r, 2 * r);
   circle.setColor(color);
   circle.setFilled(true);
   gw.add(circle, x, y);
}
```

Style Focus for Section 2:

Always Use Constants: Our code should never contain "magic numbers," meaning numbers we use in our code that don't have a clear meaning. For example don't just have 7, say **`DAYS_IN_WEEK`**. Instead of 10, we write **`EYE_RADIUS`**. Well-named constants make it clear what the purpose of the variable is, and also reduce errors. If someone wants to change the **`EYE_RADIUS`**, they can modify its value everywhere in the program by only changing it once. If we just wrote 10, they would have to go searching through the code to find all the places we use this value. The only numbers we don't need to turn into constants are the numbers 0, 1 and sometimes 2.

General and Reusable Functions: It is important to write methods that are general and reusable. If you find yourself copying and pasting code, this is probably a sign that you should have a more general method to accomplish this task. However, figuring out how to write general and reusable functions is an art, and is quite challenging. Look for similarities in your code, or ask yourself how you can use parameters.