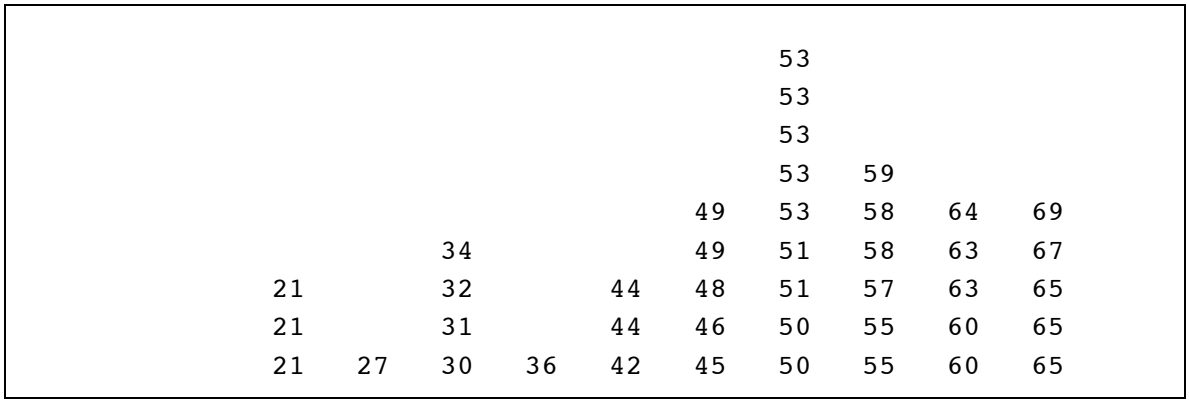
Eric Roberts and Jerry Cain
CS 106J

Handout #34
May 12, 2017

Answers to Midterm Exam

This midterm exam went well, particularly given that this is the pilot offering of CS 106J. The median was a 74.3%, which is consistent with—and perhaps even slightly higher than—historical patterns in CS 106A. The happy surprise is that there were no scores in the teens or single digits (sadly, there usually are), which means that all of you have at least some understanding of the material.

The complete histogram of grades looks like this:



You can determine your letter grade, which we've curved so that the median is at the boundary between a B+ and an A–, by looking up your score in the following table:

N = 42
Median = 52 (74.3%)
Mean = 49.3 (70.4%)

Range	Grade	N
66–70	A+	2
59–65	A	9
53–58	A–	10
50–52	B+	4
44–49	B	7
38–43	B–	1
34–37	C+	2
28–33	C	3
26–27	C–	1
20–25	D	3
00–19	NP	0

Problem 1: Karel the Robot (10 points)

```
/*
 * File: KarelHistogram.k
 * ----------------------
 * This program builds a histogram by distributing beepers upward from 1st
 * Street.  In a sense, this program is the inverse of CollectBeeperTowers.
 */

import "turns";

/*
 * Creates the histogram.  The size of the world is given in the problem.
 */

function KarelHistogram() {
   repeat (14) {
      move();
      buildColumn();
   }
   move();
}

/*
 * Builds a column by picking up the beepers and distributing them upward.
 */

function buildColumn() {
   pickAllBeepers();
   turnLeft();
   distributeBeepers();
   turnAround();
   moveToWall();
   turnLeft();
}

/*
 * Picks up all the beepers on this corner.
 */

function pickAllBeepers() {
   while (beepersPresent()) {
      pickBeeper();
   }
}

/*
 * Distributes beepers by placing one beeper on each successive corner.
 * The frontIsClear test ensures that Karel stops if it reaches the top.
 */

function distributeBeepers() {
   while (beepersInBag()) {
      putBeeper();
      if (frontIsClear()) {
         move();
      }
   }
}
```

Problem 2: Simple JavaScript expressions, statements, and functions (10 points)

(2a)

`3 + 1 * 4 - 1 + 5 + 9 % 2 + 6 * 5`	42
`"A" > "A" + "B" || "C" > "A" + "B"`	**true**
`1 + 3 + "9" + 2 * 3`	496

(2b) `"zebra"`

(2c)

JavaScript Console

```
LOL
```

Note: The call to `toLowerCase` returns the lowercase version but does not change the string, which is immutable.

Problem 3: Simple JavaScript programs (15 points)

This was the problem from the text and appears in Chapter 4 as exercise 4.

```
/*
 * File: RadioactiveDecay.js
 * -------------------------
 * This program defines the function radioactiveDecay which simulates
 * radioactive decay in a population of radioactive atoms.
 */

import "RandomLib.js";

/*
 * Simulates the radioactive decay of a population of the specified number
 * of atoms of radioactive material.  Each atom has a decayProbability
 * chance of decaying in one year.
 */

function simulateRadioactiveDecay(nAtoms, decayProbability) {
   for (var year = 1; nAtoms > 0; year++) {
      var nDecay = 0;
      for (var i = 0; i < nAtoms; i++) {
         if (randomChance(decayProbability)) {
            nDecay++;
         }
      }
      nAtoms -= nDecay;
      var str = (nAtoms === 1) ? "is 1 atom" : "are " + nAtoms + " atoms";
      console.log("There " + str + " at the end of year " + year + ".");
   }
}
```

Problem 4: Using graphics and animation (20 points)

```
/*
 * File: RollingWheel.js
 * ---------------------
 * This program rolls a wheel across the window until the wheel moves
 * offscreen.  Clicking the mouse reverses the direction of the wheel.
 */

import "graphics";

/* Constants */

const WINDOW_WIDTH = 500;
const WINDOW_HEIGHT = 300;
const WHEEL_RADIUS = 30;
const VELOCITY_X = 3;
const VELOCITY_THETA = 360 * VELOCITY_X / (2 * Math.PI * WHEEL_RADIUS);
const TIME_STEP = 20;

/* Main program */

function RollingWheel() {
   var gw = GWindow(WINDOW_WIDTH, WINDOW_HEIGHT);
   var wheel = createWheel(gw);
   gw.add(wheel, WHEEL_RADIUS, WINDOW_HEIGHT - WHEEL_RADIUS);
   var vx = VELOCITY_X;
   var vt = -VELOCITY_THETA;
   var step = function() {
      wheel.move(vx, 0);
      wheel.rotate(vt);
      if (offScreen(wheel)) clearTimeout(timer);
   };
   var timer = setInterval(step, TIME_STEP);
   var clickAction = function (e) {
      vx = -vx;
      vt = -vt;
   };
   gw.addEventListener("click", clickAction);
}

/* Creates a GCompound consisting of a circle with spokes */

function createWheel(gw) {
   var wheel = GCompound();
   wheel.add(GOval(-WHEEL_RADIUS, -WHEEL_RADIUS,
                   2 * WHEEL_RADIUS, 2 * WHEEL_RADIUS));
   wheel.add(GLine(-WHEEL_RADIUS, 0, WHEEL_RADIUS, 0));
   wheel.add(GLine(0, -WHEEL_RADIUS, 0, WHEEL_RADIUS));
   return wheel;
}

/* Returns true if the wheel has moved offscreen */

function offScreen(wheel) {
   return wheel.getX() + WHEEL_RADIUS < 0 ||
          wheel.getX() - WHEEL_RADIUS > WINDOW_WIDTH;
}
```

Problem 5: Strings (15 points)

```
/*
 * File: Portmanteau.js
 * --------------------
 * For the purposes of this problem, we're defining a portmanteau to be
 * a contraction of two words that share a vowel.  If there are no common
 * vowels, the function returns the constant null.  Otherwise, it returns
 * the contraction formed from the letters in the first word up to the
 * first occurrence of the common vowel and the letters in the second word
 * after the first occurrence of the common vowel.
 *
 * For example,
 *    portmanteau("smoke", "fog") returns "smog"
 *    portmanteau("tofu", "turkey") returns "tofurkey"
 *    portmanteau("motor", "hotel") returns "motel"
 *    portmanteau("brad", "angelina") returns "brangelina"
 *    portmanteau("spoon", "fork") returns "spork"
 */

function portmanteau(word1, word2) {
   var vp1 = findVowelAfter(word1, 0);
   while (vp1 !== -1) {
      var vp2 = word2.indexOf(word1.charAt(vp1));
      if (vp2 !== -1) {
         return word1.substring(0, vp1) + word2.substring(vp2);
      }
      vp1 = findVowelAfter(word1, vp1 + 1);
   }
   return null;
}

/*
 * Returns the index of the first vowel in the word after the
 * start position, or -1 if no more vowels exist.
 */

function findVowelAfter(word, start) {
   for (var i = start; i < word.length; i++) {
      if (isEnglishVowel(word.charAt(i))) {
         return i;
      }
   }
   return -1;
}

/*
 * Returns true if the character ch is a vowel (A, E, I, O, or U, in
 * either upper or lower case).
 */

function isEnglishVowel(ch) {
   return ch.length === 1 && "AEIOUaeiou".indexOf(ch) !== -1;
}
```