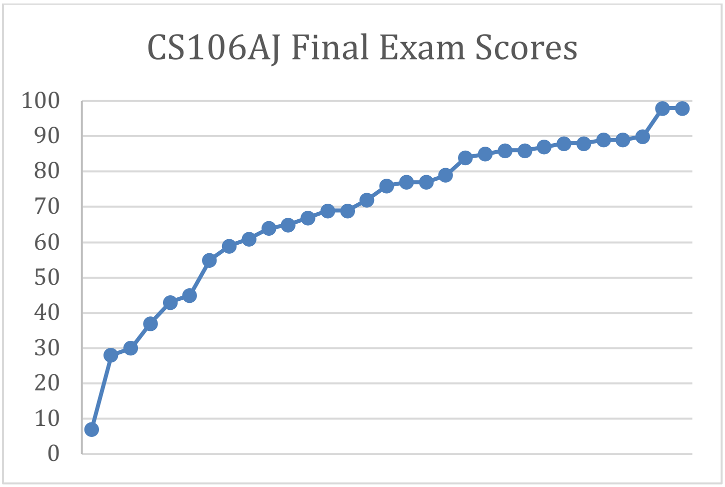
Jerry Cain
CS 106AJ

December 14, 2018
Time limit: 3 hours

CS106AJ Final Examination Solution

Your trusty CS106AJ staff cranked on Friday to press through your final exams, and I'm happy to report they're graded, and that final grades have even been computed as submitted to access. The median on the final exam was a 69 out of 100, and grades ranged from 7 all the way up to a 98.

The complete histogram of grades is presented below, where each dot represents a single exam score.



You can determine your letter grade by looking up your score in the following table:

Median = 76
Mean = 69.2

Range	Grade	N
98–100	A+	2
84–97	A	10
76–83	A–	4
72–75	B+	1
64–71	B	5
55–63	B–	3
51–54	C+	0
43–50	C	2
37–42	C–	1
28–36	D	2
00–27	NP	1

Solution 1—Short answer (10 points)

Answer to Problem 1a:

0	1	2	0	1	2	3	3	3	4

Answer to Problem 1b:

`49:8=eine`

Solution 2—Simple graphics (15 points)

```
/**
 * Function: createFilledBox
 * -------------------------
 * Construct and returns a rectangle object with the supplied
 * width, height, border color, and fill color.
 */
function createFilledBox(width, height, border, fill) {
   let box = GRect(width, height);
   box.setColor(border);
   box.setFilled(true);
   box.setFillColor(fill);
   return box;
}

/**
 * Function: Histogram
 * -----------------------
 * Creates and returns a GCompound representing a histogram of scores.
 */
function Histogram(scores) {
   let histogram = GCompound();
   let border = createFilledBox(HG_WIDTH, HG_HEIGHT, HG_BORDER, HG_FILL);
   histogram.add(border, -HG_WIDTH/2, -HG_HEIGHT/2);
   let max = findMaximum(scores);
   let w = HG_WIDTH/scores.length;
   for (let i = 0; i < scores.length; i++) {
      let scale = scores[i]/max;
      let normalized = scale * HG_HEIGHT;
      let bar = createFilledBox(w, normalized, HG_BAR_BORDER, HG_BAR_FILL);
      histogram.add(bar, i * w - HG_WIDTH/2, HG_HEIGHT/2 - normalized);
   }
   return histogram;
}
```

Solution 3—Interactive graphics (20 points)

```
/**
 * Function: MatchTheFlags
 * -----------------------
 * Constructs and plays a one-player version of MatchTheFlags.
 */
function MatchTheFlags() {
   let gw = createBoard(constructBoard());
   let selected = [];
   gw.addEventListener("click", function(e) {
      if (selected.length === 2) return;
      let flag = gw.getElementAt(e.getX(), e.getY());
      if (flag === null || flag.selected) return;
      selected.push(flag);
      flag.selected = true;
      flag.cover.setFilled(false);
      if (selected.length === 1) return;
      setTimeout(function() {
         if (selected[0].country === selected[1].country) {
            gw.remove(selected[0]);
            gw.remove(selected[1]);
         } else {
            for (let i = 0; i < 2; i++) {
               selected[i].selected = false;
               selected[i].cover.setFilled(true);
            }
         }
         selected = [];
      }, DELAY);
   });
}
```

Solution 4—Strings (15 points)

```
/**
 * Function: encrypt
 * -----------------
 * Encrypts the supplied cleartext using the supplied key.
 */
function encrypt(cleartext, key) {
   let base = "A".charCodeAt(0);
   let encrypted = "";
   for (let i = 0; i < cleartext.length; i++) {
      let keych = key.charAt(i % key.length);
      let row = ALPHABET.indexOf(cleartext.charAt(i));
      let col = TABLE[row].indexOf(keych);
      encrypted += String.fromCharCode(base + col);
   }
   return encrypted;
}
```

Solution 5—Arrays (10 points)

```
/**
 * Function: leaders
 * -----------------
 * Accepts an array of integers and returns a new array
 * containing that array's leaders.  The leaders are returned
 * in the order they appear in the original array.
 */
function leaders(array) {
   let result = [];
   for (let i = 0; i < array.length; i++) {
      let include = true;
      for (let j = i + 1; include && j < array.length; j++) {
         include = array[i] > array[j];
      }
      if (include) {
         result.push(array[i]);
      }
   }
   return result;
}
```

Solution 6—Working with data structures (15 points)

```
/**
 * Function: findUnitOfMeasure
 * ---------------------------
 * Returns the index of the record within CONVERSIONS that houses
 * information about the named unit of measure.
 */
function findUnitOfMeasure(unit) {
   for (let i = 0; i < CONVERSIONS.length; i++) {
      if (CONVERSIONS[i].unit === unit) return i;
   }
   return -1; // unit of measure isn't one to be normalized
}

/**
 * Function: normalize
 * -------------------
 * Adjusts the supplied ingredient record so that the quantity
 * is expressed in the most appropriate unit of measure.
 */
function normalize(ingredient) {
   let index = findUnitOfMeasure(ingredient.unit);
   if (index === -1) return;
   for (let i = index - 1; i >= 0; i--) {
      ingredient.amount *= CONVERSIONS[i].amount;
   }
   ingredient.unit = CONVERSIONS[0].unit;
   for (let i = 0; i < CONVERSIONS.length - 1; i++) {
      if (ingredient.amount < CONVERSIONS[i].amount) break;
      ingredient.amount /= CONVERSIONS[i].amount;
      ingredient.unit = CONVERSIONS[i + 1].unit;
   }
}
```

```
/**
 * Function: scale
 * ---------------
 * Scales the recipe so that the quantities expressed are enough for
 * precisely ‘servings’ servings.
 */
function scale(recipe, servings) {
   let scale = servings/recipe.servings;
   for (let i = 0; i < recipe.ingredients.length; i++) {
      recipe.ingredients[i].amount *= scale;
      normalize(recipe.ingredients[i]);
   }
   recipe.servings = servings;
}
```

Solution 7— Reading data structures from embedded XML (15 points)

```
/**
 * Function: buildCodebook
 * -----------------------
 * Extracts the XML needed to configure an object that maps dates
 * to Enigma machine configurations.
 */
function buildCodebook() {
   let codebookXML = document.getElementById("codebook");
   if (codebookXML === null) return undefined; // not required of students
   let codebook = {};
   let entriesXML = codebookXML.getElementsByTagName("entry");
   for (let i = 0; i < entriesXML.length; i++) {
      let entryXML = entriesXML[i];
      let date = entryXML.getAttribute("date");
      let config = {};
      config.order = entryXML.getAttribute("order");
      config.setting = entryXML.getAttribute("settings");
      config.pairings = [];
      let pairingsXML = entryXML.getElementsByTagName("pairing");
      for (let j = 0; j < pairingsXML.length; j++) {
         let pairingXML = pairingsXML[i];
         config.pairings.push(pairingXML.getAttribute("value"));
      }
      codebook[date] = config;
   }
   return codebook;
}
```