Answers to Midterm Exam

The course staff spent several hours grading CS106AJ midterms on Saturday morning and early afternoon, and I’m happy to report that they’re graded and will be returned during lecture on Monday. The exam was intended to be challenging, but many of you did brilliantly, and most of you did well enough that I’m happy to go with a traditional CS106A curve, where I set the median grade to sit at the A-/B+ border.

The complete histogram of grades is presented below, where each vertical bar represents a single exam score, and the higher scores on the left and lower scores are on the right.

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:

<table>
<thead>
<tr>
<th>Range</th>
<th>Grade</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>64–70</td>
<td>A+</td>
<td>4</td>
</tr>
<tr>
<td>49–62</td>
<td>A</td>
<td>14</td>
</tr>
<tr>
<td>40–48</td>
<td>A–</td>
<td>8</td>
</tr>
<tr>
<td>35–39</td>
<td>B+</td>
<td>4</td>
</tr>
<tr>
<td>27–34</td>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>23–26</td>
<td>B–</td>
<td>5</td>
</tr>
<tr>
<td>20–22</td>
<td>C+</td>
<td>1</td>
</tr>
<tr>
<td>17–19</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>15–16</td>
<td>C–</td>
<td>1</td>
</tr>
<tr>
<td>13–14</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>00–12</td>
<td>NP</td>
<td>0</td>
</tr>
</tbody>
</table>

Median = 40 (57.1%)
Mean = 40.5 (57.9%)
Solution 1: Karel the Robot (10 points)

```javascript
function CalendarKarel() {
  advanceToBeepers();
  countDaysInMonth();
  constructCalendar();
}

function advanceToBeepers() {
  while (noBeepersPresent()) {
    move();
  }
}

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

function constructCalendar() {
  while (beepersInBag()) {
    constructWeek();
  }
}
```

* Program: CalendarKarel
  * ---------------------
  * Manages the construction of an entire calendar month,
  * which amounts to:
  *
  * + getting Karel to advance to the beeper pile,
  * + collect all of those (between 28 and 31) beepers, and
  * + lay down all of the beepers across several weeks
  * until Karel is out of beepers
  */
/**
 * Function: constructWeek
 * -----------------------
 * We know that constructWeek can only get called
 * if there's at least one beeper in Karel's bag,
 * so Karel definitely puts one beeper down. Then
 * Karel is prepared to move and put another beeper
 * down up to six more times, though stops doing
 * that if there are no more beepers in Karel's bag (month
 * ends on a day other than Saturday) and/or there's a
 * wall (because the beginning of the month fell after
 * a Sunday).
 *
 * Once the current week has been constructed, Karel
 * prepares to construct the next week.
 */
function constructWeek() {
    putBeeper();
    repeat (6) {
        if (beepersInBag()) {
            if (frontIsClear()) {
                move();
                putBeeper();
            }
        }
    }
    if (beepersInBag()) {
        positionForNextWeek();
    }
}

/**
 * Function: positionForNextWeek
 * -----------------------------
 * Rewinds Karel to the left margin of the
 * calendar and then drops Karel down to the
 * beginning of the next week, leaving Karel
 * facing east.
 */
function positionForNextWeek() {
    turnAround();
    moveToWall(); // as written in lecture and in the course reader
    turnLeft();
    move();
    turnLeft();
}
Solution 2: Simple JavaScript expressions, statements, and functions (10 points)

(2a) \[ 5 + 7 \times 7 - 2 + 1 + 5 \% 6 + 6 \times 4 \% 9 \]

\[ "C" === "CC" \| \"Y" < \"X" + \"Y" \]

\[ 7 + 1 + "1" + 4 \times 7 \]

---

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 7 * 7 - 2 + 1 + 5 % 6 + 6 * 4 % 9</td>
<td>64</td>
</tr>
<tr>
<td>&quot;C&quot; === &quot;CC&quot;</td>
<td>false</td>
</tr>
<tr>
<td>&quot;Y&quot; &lt; &quot;X&quot; + &quot;Y&quot;</td>
<td></td>
</tr>
<tr>
<td>7 + 1 + &quot;1&quot; + 4 * 7</td>
<td>&quot;8128&quot;</td>
</tr>
</tbody>
</table>

(2b) "honesnesn"

(2c) "thisgoods"

Note: The call to toUpperCase returns the uppercase version but does not change the string, which is immutable.

Solution 3: Simple JavaScript programs (15 points)

```javascript
/**
 * Function: isValid
 * -----------------
 * Returns true if and only if the supplied number
 * meets the requirements imposed by Luhn’s algorithm.
 */
function isValid(number) {
    let sum = 0;
    let double = false;
    while (number > 0) {
        let digit = number % 10;
        if (double) digit *= 2;
        if (digit >= 10) digit -= 9;
        sum += digit;
        number = Math.floor(number/10);
        double = !double;
    }
    return sum % 10 === 0;
}
```
Solution 4: Using graphics and animation (20 points)

```javascript
/**
 * File: Quadrilaterals.js
 * -----------------------
 * Defines the graphics program that allows a user to place an
 * unlimited number of quadrilaterals in the graphics window.
 * Each quadrilateral is really a GPolygon, and the four vertices
 * of each GPolygon are defined by a sequence of four mouse clicks.
 *
 * Each quadrilateral, as it's added to the graphics window,
 * is outlined in black, filled with a random color, and
 * configured to change colors every five seconds.
 */
function Quadrilaterals() {
    let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
    let inProgress = null;
    let numClicks = 0;
    let clickAction = function(e) {
        if (numClicks === 0) inProgress = GPolygon();
        placeDot(gw, e.getX() - DOT_DIAMETER/2,
                 e.getY() - DOT_DIAMETER/2);
        inProgress.addVertex(e.getX(), e.getY());
        numClicks++;

        if (numClicks < 4) return;

        inProgress.setFilled(true);
        inProgress.setFillColor(randomColor());
        gw.add(inProgress);

        let completed = inProgress;
        setInterval(function() {
            completed.setFillColor(randomColor());
        }, TIME_STEP);
        numClicks = 0;
    }
    gw.addEventListener("click", clickAction);
}

/**
 * Function: placeDot
 * ------------------
 * Places a small black dot so that its center overlays the
 * the supplies (cx, cy) coordinate.
 */
function placeDot(gw, cx, cy) {
    let dot = GOval(cx, cy, DOT_DIAMETER, DOT_DIAMETER);
    dot.setFilled(true);
    gw.add(dot);
}
```
Solution 5: Strings (15 points)

```javascript
/**
 * Function: translate
 * -------------------
 * Translates the supplied word into the equivalent word in B-Language, as outlined in the Problem 5 statement.
 */
function translate(word) {
    let translation = "";
    let start = -1;
    for (let i = 0; i < word.length; i++) {
        let ch = word.charAt(i);
        if (isEnglishVowel(ch)) {
            if (start === -1) {
                start = i;
            }
        } else {
            if (start !== -1) {
                let cluster = word.substring(start, i);
                translation += cluster + 'b' + cluster;
                start = -1;
            }
            translation += ch;
        }
    }
    if (start !== -1) {
        let cluster = word.substring(start);
        translation += cluster + 'b' + cluster;
    }
    return translation;
}

/**
 * Function: isEnglishVowel
 * ------------------------
 * Returns true if and only if the supplied string is a single-character string, and that one character is a lowercase vowel.
 */
function isEnglishVowel(ch) {
    return ch.length === 1 && "aeiou".indexOf(ch) >= 0;
}
```