Problem 1: Karel the Robot (10 points)

```java
/*
 * File: KarelCare
 * -------
 * Karel looks through the hospital ward for patients with
 * temperatures over 100 and paints the square under the
 * temperature red so that doctors can treat the patient.
 */

import "turns";
import "extensions";

/* Main program */

function KarelCare() {
    while (frontIsClear()) {
        if (beepersPresent()) {
            checkTemperature();
        }
        move();
    }
    if (beepersPresent()) {
        checkTemperature();
    }
}

/* Flags temperatures greater than 100 */

function checkTemperature() {
    repeat (100) {
        if (beepersPresent()) {
            pickBeeper();
        }
    }
    if (beepersPresent()) {
        paintCorner("Red");
    } while (beepersInBag()) {
        putBeeper();
    }
}
```
Problem 2: Simple JavaScript expressions, statements, and functions (10 points)

(2a) \[ 10 \times 9 + 8 \times 7 \times 6 \times 5 + 4 \times 3 / 2 / 1 = 1776 \]

```javascript
let x = 7;
(x !== 6) || (x !== 7) // true

"E".charCodeAt(0) - "A".charCodeAt(0) // 4
```

(2b) 1 (This program calculates the digital root as described in Chapter 4, exercise 8.)

(2c)

<table>
<thead>
<tr>
<th>JavaScript Console</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1 = Heart</td>
</tr>
<tr>
<td>s2 = earth</td>
</tr>
</tbody>
</table>

Problem 3: Simple JavaScript programs (15 points)

```javascript
/**
 * Function: consecutiveHeads
 * --------------------------
 * Simulates the process of flipping a fair coin until the number of
 * consecutive heads matches the number supplied via numHeadsNeeded.
 */
function consecutiveHeads(numHeadsNeeded) {
  let numTosses = 0;
  let numHeads = 0;
  while (numHeads < numHeadsNeeded) {
    if (randomChance()) { // let true mean heads
      console.log("Heads");
      numHeads++;
    } else {
      console.log("Tails");
      numHeads = 0;
    }
    numTosses++;
  }
  console.log("It took "+ numTosses + " tosses to get "+ numHeadsNeeded + " consecutive heads.");
}
```
Problem 4: Using graphics and animation (20 points)

    /*
     * File: Fireworks.js
     * -----------------
     * The program animates the launch and explosion of a single firework,
     * as per the Problem 4 problem statement in Practice Midterm 2.
     */

    /* Constants (in pixels) */
    const GWINDOW_WIDTH = 500;
    const GWINDOW_HEIGHT = 300;
    const DELTA_RADIUS = 2;

    /* Constants (in milliseconds) */
    const TIME_STEP = 20;
    const FLIGHT_TIME = 1200;
    const EXPANSION_TIME = 500;

    /* Derived Constants */
    const TOTAL_TIME = FLIGHT_TIME + EXPANSION_TIME; /* in milliseconds */
    const NUM_STEPS = FLIGHT_TIME / TIME_STEP;

    /* Main program */
    function Fireworks() {
        let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
        let radius = 1;
        let firework = GOval(gw.getWidth()/2, gw.getHeight(),
            radius, radius);
        firework.setColor(randomColor());
        let targetx = randomReal(0, gw.getWidth());
        let targety = randomReal(0, gw.getHeight()/2);
        let dx = (targetx - firework.getX()) / NUM_STEPS;
        let dy = (targety - firework.getY()) / NUM_STEPS;
        let t = 0;
        gw.add(firework);
        let step = function() {
            if (t < FLIGHT_TIME) {
                firework.move(dx, dy);
            } else if (t < TOTAL_TIME) {
                radius += DELTA_RADIUS;
                firework.setBounds(firework.getX() - DELTA_RADIUS,
                    firework.getY() - DELTA_RADIUS,
                    2 * radius, 2 * radius);
            } else {
                clearInterval(timer);
            }
            t += TIME_STEP; // time advances no matter what happened
        }
        let timer = setInterval(step, TIME_STEP);
    }
Problem 5: Strings (15 points)
/**
 * File: Portmanteau.js
 * ---------------------
 * Defines the portmanteau function according to the specifications
 * laid out in the final problem of the second practice midterm.
 */

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

/**
 * Function: findFirstVowel
 * ------------------------
 * Returns the index of the first lowercase vowel at or after
 * the provided start position, or -1 if no lowercase vowel
 * could be found. If the call to findFirstVowel omitted the
 * second parameter, then start is assumed to be 0.
 */
function findFirstVowel(word, start) {
    if (start === undefined) start = 0;
    for (let i = start; i < word.length; i++) {
        if (isEnglishVowel(word.charAt(i))) {
            return i;
        }
    }
    return -1;
}

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