Solutions to Practice Midterm #2

Tuesday, October 29, 3:30–5:30 P.M., 420-041
Tuesday, October 29, 7:00–9:00 P.M., 420-401

Solution 1: Simple JavaScript expressions, statements, and functions (10 points)

(1a) \(3 + 2 \times 2 - 15 \% 5 \times 100\)

\(\text{false}\)

(1b) "mior"

(1c) "IGHOWEEN24"

Solution 2: Using graphics and animation (15 points)

```javascript
/* 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;

// full program using above constants is on the next page
```
/ * 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);
}
Solution 3: Strings (15 points)

```javascript
/**
 * File: Portmanteau.js
 * -------------------
 * Defines the portmanteau function according to the specifications
 * laid out in the third 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 character is a lowercase vowel.
 */
function isEnglishVowel(ch) {
    return ch.length === 1 && "aeiou".indexOf(ch) >= 0;
}
```
Solution 4: Arrays (15 points)

```javascript
/**
 * Function: dedupe
 * ------------
 * Updates the supplied array such that all duplicates
 * are removed. The implementation is designed to work
 * for arrays of any single primitive type (e.g. an array
 * of numbers, or an array of strings, or an array of bools)
 */
function dedupe(array) {
    for (let i = array.length - 1; i >= 0; i--) {
        if (array.indexOf(array[i]) < i) {
            array.splice(i, 1);
        }
    }
}
```

Solution 5: Working with data structures (15 points)

```javascript
/**
 * Function: facebookRefund
 * ------------------------
 * Decides whether it was less expensive to purchase
 * Facebook stock at the time an order was placed or
 * the time the trade was executed and returns the
 * price difference between the two if the latter was
 * less expensive (and 0 otherwise).
 */
function facebookRefund(nShares, date, timeOrdered, timeExecuted) {
    let priceOrdered = findSharePrice(date, timeOrdered);
    let priceExecuted = findSharePrice(date, timeExecuted);
    let refund = nShares * (priceOrdered - priceExecuted);
    if (refund < 0) refund = 0;
    return refund;
}

/**
 * Function: findSharePrice
 * ------------------------
 * Returns the price of Facebook stock at the specified
 * time on the specified date. If no price information is
 * available, an alert notifies the user and 0.0 is returned.
 */
function findSharePrice(date, time) {
    for (let i = 0; i < FB_SHARE_PRICE_DATA.length; i++) {
        let entry = FB_SHARE_PRICE_DATA[i];
        if (entry.date === date && entry.time === time) {
            return entry.price;
        }
    }
    alert("No record for " + date + " " + time + ".");
    return 0.0;
}
```