Jerry Cain
CS 106AJ

Handout #47S
November 30, 2018

Answers to Practice Final #2

Review session: Saturday, December 8, 11:00A.M. –1:00P.M. (McCullough 115)
Scheduled final: Monday, December 10, 8:30–11:30A.M. (380-380C)

Problem 1—Short answer (10 points)

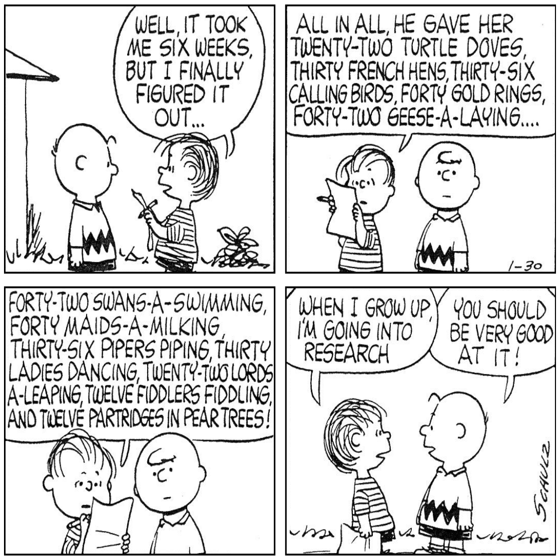
1a) **`array`**

0	12	22	30	36	40	42	42	40	36	30	22	12
0	1	2	3	4	5	6	7	8	9	10	11	12

The values for this problem (which was taken from an autumn quarter final and thus came at the right season) indicate the total number of gifts in each category if you take the words to "The Twelve Days of Christmas" literally. At the end of our true love's gift-giving spree, the total haul contains:

12 Partridges in pear trees
22 Turtle doves
30 French hens
36 Calling birds
40 Gold rings
42 Geese-a-laying
42 Swans-a-swimming
40 Maids-a-milking
36 Ladies dancing
30 Lords-a-leaping
22 Pipers piping
12 Drummers drumming

Charles M. Schulz offered a lovely rendition of this problem in 1963:


WELL, IT TOOK ME SIX WEEKS, BUT I FINALLY FIGURED IT OUT...
ALL IN ALL, HE GAVE HER TWENTY-TWO TURTLE DOVES, THIRTY FRENCH HENS, THIRTY-SIX CALLING BIRDS, FORTY GOLD RINGS, FORTY-TWO GEESE-A-LAYING....
1-30
FORTY-TWO SWANS-A-SWIMMING, FORTY MAIDS-A-MILKING, THIRTY-SIX PIPERS PIPING, THIRTY LADIES DANCING, TWENTY-TWO LORDS A-LEAPING, TWELVE FIDDLERS FIDDLING, AND TWELVE PARTRIDGES IN PEAR TREES!
WHEN I GROW UP, I'M GOING INTO RESEARCH
YOU SHOULD BE VERY GOOD AT IT!
SCHULZ

1b) The issue here is figuring out exactly which variable or field each occurrence of **`x`** refers to. The answer is the string consisting of `"" + 10 + (11 * 6)`, or `"1066"`.

Problem 2—Simple graphics (15 points)

```
/**
 * Function: createButton
 * ----------------------
 * Creates a button (as a GCompound) to surround the specified
 * text, as specified in the problem statement.  The reference
 * point of the button is the upper left corner of the smallest
 * rectangle circumscribing the button rendering.
 */
function createButton(text) {
   let button = GCompound();
   let label = GLabel(text);
   let radius = BUTTON_HEIGHT/2;
   let diameter = 2 * radius;
   label.setFont(BUTTON_FONT);
   button.add(GLine(radius, 0, radius + label.getWidth(), 0));
   button.add(GArc(label.getWidth(), 0, diameter, diameter, 270, 180));
   button.add(GLine(radius, diameter, radius + label.getWidth(), diameter));
   button.add(GArc(0, 0, diameter, diameter, 90, 180));
   button.add(label, radius, radius + BUTTON_LABEL_DY);
   return button;
}
```

Problem 3—Interactive graphics (20 points)

```
/* Derived constants */
const COIN_SEP = (GWINDOW_WIDTH - N_COINS * COIN_SIZE)/(N_COINS + 1);

/**
 * Function: GraphicNim
 * --------------------
 * Defines the factory function that manages the entire
 * simulation.
 */
function GraphicNim() {
   let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
   let coins = createCoinsArray(gw);
   let clickAction = function (e) {
      let coin = gw.getElementAt(e.getX(), e.getY());
      if (coin === null) return;
      let pos = coins.indexOf(coin);
      if (pos < coins.length - 3) return;
      let count = coins.length - pos;
      for (let i = pos; i < coins.length; i++) {
         gw.remove(coins[i]);
      }
      coins.splice(pos, count);
   };
   gw.addEventListener("click", clickAction);
}

/**
 * Function: createCoinsArray
 * --------------------------
 * createCoinsArray constructs and returns an array of N_COINS
 * GOvals to represent each of the coins.  Each coin is properly
 * set in the supplied graphics window so that the array of
 * coins is centered both vertically and horizontally.
 */
function createCoinsArray(gw) {
   let coins = [];
   let y = (gw.getHeight() - COIN_SIZE)/2;
   for (let i = 0; i < N_COINS; i++) {
      let x = COIN_SEP + i * (COIN_SIZE + COIN_SEP);
      let coin = GOval(x, y, COIN_SIZE, COIN_SIZE);
      coin.setFilled(true);
      coin.setFillColor(COIN_FILL_COLOR);
      gw.add(coin);
      coins.push(coin);
   }
   return coins;
}
```

Problem 4—Strings (15 points)

```
/* Constants */
const TOC_LINE_LENGTH = 60;

/**
 * Function: createTocEntry
 * ------------------------
 * Returns a string of length TOC_LINE_LENGTH that's prefixed by
 * title, suffixed by the page number, and includes a leader in
 * between the title and page number according to the rules laid
 * out in the problem statement.
 */
function createTocEntry(title, page) {
   if (title.length % 2 === 0) title += " ";
   let entry = title + " ";
   page = " " + page; // convert page from number to string with padding
   let gap = TOC_LINE_LENGTH - entry.length - page.length;
   let leader = "";
   for (let i = 0; i < gap; i++) {
      let ch = i % 2 === 0 ? "." : " ";
      leader += ch;
   }
   return entry + leader + page;
}
```

Problem 5—Arrays (10 points)

```
/**
 * Function: rotateArray
 * ---------------------
 * Simple function that removes the first
 * k elements of an array and appends them to the
 * end, for any nonnegative value of k less than
 * or equal to the array length.  The function
 * doesn't return anything, but rather relies on
 * the fact that the array is shared by reference and
 * thus updated in place.
 */
function rotateArray(array, k) {
   for (let i = 0; i < k; i++) {
      array.push(array.shift());
   }
}
```

Problem 6—Working with data structures (15 points)

```
/**
 * 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;
}
```

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

```
/**
 * Function: readLetters
 * ---------------------
 * Parses the XML content of the surrounding HTML file and returns
 * an array of aggregates, where each aggregate stores information
 * about a single exchange between two people, as specified in the
 * problem statement.
 */
function readLetters() {
   let exchangesXML = document.getElementById("ExchangesData");
   let letterElements = exchangesXML.getElementsByTagName("letter");
   let letters = [];
   for (let i = 0; i < letterElements.length; i++) {
      letters.push(letterElementToAggregate(letterElements[i]));
   }
   return letters;
}

/**
 * Function: letterElementToAggregate
 * ----------------------------------
 * Parses the XML rooted at the provided letter element
 * and returns an aggregate housing precisely the same
 * information, as outlined in the problem statement.
 */
function letterElementToAggregate(element) {
   let lines = element.innerHTML.split("\n");
   let body = [];

   for (let i = 0; i < lines.length; i++) {
      let line = lines[i].trim();
      if (line.length > 0) body.push(line);
   }

   let letter = {
      to: element.getAttribute("to"),
      from: element.getAttribute("from"),
      date: element.getAttribute("date"),
      body: body
   };

   return letter;
}
```