Jerry Cain
CS 106AJ

October 30, 2018
Time limit: 2 hours

Midterm Examination

Name (*please print*) ________________________________

Section Leader ________________________________

	score	*initials*
1	_____/10	____
2	_____/10	____
3	_____/15	____
4	_____/20	____
5	_____/15	____
Total	_____/70	

General instructions

Answer each of the five questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit.

Each question is marked with the number of points assigned to that problem. The total number of points is 70. We intend for the number of points to be roughly comparable to the number of minutes you should spend on that problem. This leaves you with an additional 50 minutes to check your work or recover from false starts.

In all questions, you may include methods or definitions that have been developed in the course, either by writing the **`import`** line for the appropriate Karel package, or by giving the name of the method and the handout or chapter number in which that definition appears.

Unless otherwise indicated as part of the instructions for a specific problem, comments will not be required on the exam. Uncommented code that gets the job done will be sufficient for full credit on the problem. On the other hand, comments may help you to get partial credit if they help us determine what you were trying to do.

The examination is open-book, and you may make use of any texts, handouts, or course notes. You may not, however, use a computer of any kind.

THE STANFORD UNIVERSITY HONOR CODE

A. The Honor Code is an undertaking of the students, individually and collectively:
  (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid as far as practicable, academic procedures that create temptations to violate the Honor Code.

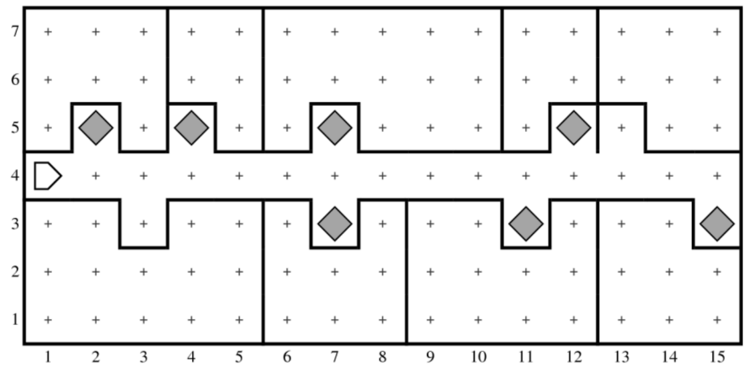
C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

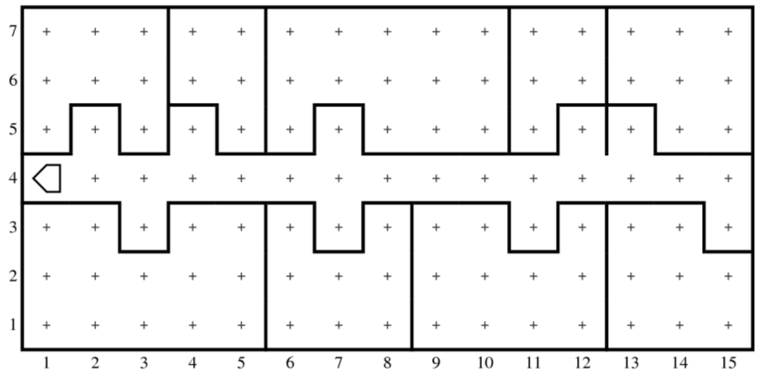
(signed) ____

Problem 1: Karel the Robot (10 points)

Suppose that it's Halloween, and Karel is going Trick-or-Treating. Karel starts off at the west end of a dead-end street that contains houses on both sides, such as the one pictured in the following diagram:



Each house has a front porch at some point along its front side. Karel's mission is to go to each house, step into the porch area, and see if the porch contains a treat, represented by a beeper. If it does, Karel should pick up the treat, and if not, Karel should move on to the next house. Karel must check every porch on both sides of the street and should end up at the original intersection, facing in the opposite direction. Thus, after executing your program in the world shown above, Karel should finish in the following position:



Karel can count on the following facts about the world:

- Karel starts at the west end of a street, facing east, with an empty beeper bag.
- The houses on the street are packed closely together, with no space between adjacent houses.
- There may be any number of houses, which typically vary in size.
- The side of each house facing the street is a solid wall except for a small porch, which is always one intersection wide. The porch can appear at any point in the front wall.
- Each porch is either empty or marked with a single beeper representing a Halloween treat.

- At the end of execution, Karel should return to its original intersection at the west end of the street but should now be facing west.

Use the rest of this page and the next to present your implementation of the Halloween program. You are strongly encouraged to decompose, writing helper functions and referencing existing ones that appear anywhere the course reader or in any course handout.

(space for the answer to problem #1 also appears on the next page)

Answer to Problem 1 (continued):

Problem 2: Simple Java expressions, statements, and methods (10 points)

(2a) Compute the value of each of the following JavaScript expressions:

```
3 + 2 * 2 - 15 % 5 * 100          ______________________

"B" === "b" || "H" < "GGG"        ______________________

20 + 7 + "1" + 8 + 4 * 7          ______________________
```

(2b) Assume that the method `riddle` has been defined as follows:

```
function riddle(str) {
  let result = "";
  for (let i = 0; i < str.length; i++) {
     if (i === str.lastIndexOf(str.charAt(i))) {
        result += str.charAt(i);
        str = str.substring(i + 1);
        i = -1;
     }
  }
  return result;
}
```

What is the value returned by `riddle("mirrormirror")`?

Answer to problem 2b: ______________________

(2c) What output is printed by the following `Problem2c` program?

```
function Problem2c() {
   let day = "halloween";
   let fn = function(x, y, z) {
      return z.substring(x, y) + day.substring(y);
   };
   day = spooky(fn, day.indexOf("a"), day.indexOf("o"));
   day.toLowerCase();
   console.log(day);
}

function spooky(f, x, y) {
   let ghost = f(x, y, "nightmare");
   ghost += "xyz".charCodeAt(1) - "a".charCodeAt(0);
   return ghost.toUpperCase();
}
```

Answer to problem 2c:

Problem 3: Simple JavaScript programs (15 points)

A n-digit number is considered narcissistic if and only if it's equal to the sum of each of its digits raised to the nth power. For example, 153, 370, 371, and 407 are all three-digit narcissistic numbers, because all of the following are true:

$$1^3 + 5^3 + 3^3 \text{ equals } 153$$
$$3^3 + 7^3 + 0^3 \text{ equals } 370$$
$$3^3 + 7^3 + 1^3 \text{ equals } 371$$
$$4^3 + 0^3 + 7^3 \text{ equals } 407$$

To be clear, numbers are very rarely narcissistic, since it's unusual for an n-digit number to incidentally equal to the sum of its digits raised to the nth power. In fact, the four numbers listed above are the only three-digit narcissistic numbers.

There are a several more, though. The numbers 1 through 9, inclusive, are all (trivially) narcissistic. More interesting examples include 9474, 54748, and 88593477. They're each narcissistic because each of the following statements is true:

$$9^4 + 4^4 + 7^4 + 4^4 \text{ equals } 9474$$
$$5^5 + 4^5 + 7^5 + 4^5 + 8^5 \text{ equals } 54748$$
$$8^8 + 8^8 + 5^8 + 9^8 + 3^8 + 4^8 + 7^8 + 7^8 \text{ equals } 88593477$$

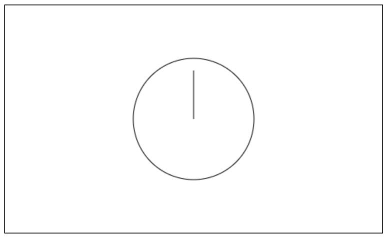
Using the space on the next page, write a program called **`Narcissistic`** that accepts a positive integer and returns **`true`** if the provided number is narcissistic according to the above description, and **`false`** otherwise. You're encouraged to decompose your solution and rely on a helper function—one you also need to implement—that counts the number of digits in a number.

(space for the answer to problem #3 appears on the next page)

Answer to problem #3

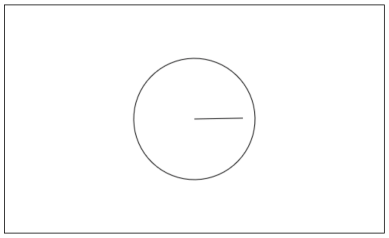
```
function Narcissistic(number) {
```

Problem 4: Using graphics and animation (20 points)
Write a simple graphics program that places a stopwatch so that it's centered within the 500px x 300px screen, as with:



The stopwatch must be implemented as a **`GCompuound`**, where a circle of radius **`STOPWATCH_RADIUS`** is centered around the reference point, and the stopwatch's hand extends upwards from the circle's center.

Your graphics program should respond to mouse clicks, where alternating mouse clicks start and stop the stopwatch. Restated, the stopwatch is stationary until the user clicks the mouse anywhere in the graphics window, at which point the stopwatch's hand rotates at a speed of 360 degrees per minute. When the user clicks a second time, the stopwatch stops, but a third click prompts it to continue as if never interrupted. The fourth click would stop again, and so on, and so on. So, if I click once and then after 15 seconds click again, the above graphic would rotate into:



The animation can, of course, be implemented using timer functions, such that each animation step rotates the hand precisely one degree clockwise every 36000/360 milliseconds. And because the problem is constraining the reference point to be the stopwatch's center, you can rotate the hand by just rotating the entire stopwatch. Circles are good about staying in place when you rotate them about their center. ☺

Use the space on the next page to present your entire **`Stopwatch`** program.

(space for the answer to problem #4 appears on the next page)

Answer to problem #4:

```
/* Constants */
const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 300;
const STOPWATCH_RADIUS = 80;
const STEP_TIME = 60000/360; // rotate through 360 degrees every 60 seconds
const DTHETA = 1;

/* Derived Constants */
const STOPWATCH_DIAMETER = 2 * STOPWATCH_RADIUS;
const STOPWATCH_HANDLENGTH = 0.8 * STOPWATCH_RADIUS; // 80% of radius

function Stopwatch() {
```

Problem 5: Strings (15 points)

Implement the **`exchange`** function, which accepts a string (one you can assume is comprised of lowercase letters and nothing else) and returns a new string that's the same as the original, save for the fact that all vowels have been pairwise exchanged. Restated, **`exchange`** returns a new string where the first and second vowels have been swapped, the third and fourth vowels have been swapped, and so forth. If there are an odd number of vowels, then the last vowel stays put. And if the supplied string has no vowels at all, then exchange returns a verbatim copy of the original.

Here are some examples of how **`exchange`**, which you'll implement to examine English words comprised of only lowercase letters.

```
         exchange("abbey") ⇒ "ebbay"
  exchange("acquaintance") ⇒ "ucqaiantenca"
 exchange("embellishment") ⇒ "embelleshmint"
     exchange("seriously") ⇒ "sireuosly"
         exchange("think") ⇒ "think"
           exchange("xyz") ⇒ "xyz"
              exchange("") ⇒ ""
```

Use the next page to present your implementation of **`exchange`**.

(space for the answer to problem #5 appears on the next page)

Answer to problem #5:

```
function exchange(str) {
```