Second Exam
CS 2254 Web App Development

KEY

Wednesday April 5, 2017

Instructor Muller
Boston College
Spring 2017

Before reading further, please arrange to have an empty seat on either side of you. Now that you are seated, please write your name **on the back** of this exam.

This is a closed-notes and closed-book exam. Computers, calculators, and books are prohibited.

- Partial credit will be given so be sure to show your work.
- Feel free to write helper functions if you need them.
- **Please write neatly.**

Problem	Points	Out Of
1		6
2		14
Total		20

1. (6 Points): What Happens with this JS Code?

What is the result of running each of these? If there is an error, what is the problem? If something is logged, what is logged?

1.
   ```
   let a = [1, 2, 3]

   a[0] = 4
   console.log(a)

   a = [5, 6]
   console.log(a)
   ```

2.
   ```
   const c = [1, 2, 3]

   c[0] = 4
   console.log(c)

   c = [5, 6]
   console.log(c)
   ```

3.
   ```
   let darren = {name: "Darren"};
   darren.age = 22;
   console.log("Darren = ", darren);
   ```

4.
   ```
   function f(n) {
     if (n === 0) return;
     console.log("f: n =", n);
     f(n - 5);
   }
   f(4);
   ```

5.
   ```
   let d = "BC " + 4;
   console.log(d);
   ```

6.
   ```
   let e = "BC " * 4;
   console.log(e);
   ```

Answer:

```
1. [4, 2, 3]
   [5, 6]
2. [4, 2, 3]
   error: assigning to a constant
3. Darren = {name: "Darren", age: 22}
4. logs f: n = 4, f: n = -1, ..., then stack overflow
5. "BC 4"
6. NaN
```

2. (14 Points): JavaScript Coding

1. (4 Points) There are 24 * 60 = 1440 minutes in a day. Write a JavaScript function `time(n)` such that for $0 \leq \mathbf{n} < 1$, `time(n)` returns a record of the form

   ```
   { hour : number, minutes : number, meridiem : string }
   ```

   For example, the call `time(.45)` would return the record

   ```
   { hour : 10, minutes : 48, meridiem : "AM" }
   ```

   Your time should be in 12-hour clock time. Feel free to use the `Math.floor` function and/or the integer remainder (mod) operator `%` in your solution.

   Answer:

   ```
   function time(n) {
       const mins = Math.floor(n * (24 * 60));
       let minutes = mins % 60;
       let hour = Math.floor(mins / 60);
       let meridiem = hour < 12 ? "AM" : "PM";
       hour = hour < 12 ? hour : hour - 12;
       hour = hour === 0 ? 12 : hour;
       return {hour, minutes, meridiem};
   }
   ```

2. (4 Points) Write a JavaScript function

```
every : A list * (A -> bool) -> bool
```

such that a call `every(xs, test)` returns `true` if and only if `test(x)` is `true` for every element of `xs`. Otherwise `every` should return `false`. For example, the call

```
every([1, 3, 5], n => n % 2 === 1)
```

should return `true` as should `every([], n => n % 2 === 1)`.

Answer:

```
function every(xs, test) {
  let answer = true;
  for(let i = 0; i < xs.length; i++)
    answer = answer && test(xs[i]);
  return answer;
}
```

3. (6 Points) Let's say we have a *binary tree* `btree` that is either a number or a record of the form `{info : number, left : btree, right : btree}`. For example, the JS forms:

```
let t1= {info: 5,         let t2= {info: 5,         let t3= {info: 5,
         left: {info: 4,           left: {info: 3,           left: {info: 3,
                left: 2,                  left: 2,                  left: 2,
                right: 1},                right: 1},                right: 1},
         right: 3}                 right: 8}                 right: {info: 6,
                                                                   left: 6,
                                                                   right: 6}}
```

would represent (resp.) the binary trees:

```
       t1 = 5               t2 = 5                t3 = 5
           / \                  / \                  /   \
          4   3                3   8                3     6
         / \                  / \                  / \   / \
        2   1                2   1                2   1 6   6
```

Let's further assume the existence of a function `isNode : btree -> bool` that returns `true` if a btree is a record and not a number. Write a JS function `diff : btree list -> btree list` such that a call `diff(btrees)` returns a list of subtrees where not all of `btrees` agree. For example, the call `diff([t1, t2, t3])` would return the list of trees

```
   [ 4,     3,     3 ]
    / \    / \    / \
   2   1  2   1  2   1
```

the call `diff([t1, t1])` would return the empty list `[]` and the call `diff([t2, t3])` would return the list of trees:

```
   [ 8,     6 ]
           / \
          6   6
```

Answer here.

Answer:

```
let rootVal = tree => (typeof(tree) === 'number' ? tree : tree.info);

let sameRoot = (t1, t2) =>
    (typeof(t1) === typeof(t2) && rootVal(t1) === rootVal(t2));

let allSameRoots = trees => trees.every(tree => sameRoot(tree, trees[0]));

let differRoots = trees => (trees.length >= 2 && !allSameRoots(trees));

function diff(trees) {
    if (differRoots(trees)) return trees;

    if (typeof(trees[0]) === 'number') return [];

    else {
        let lefts = trees.map(tree => tree.left);
        let leftDiffs = diff(lefts);
        if (leftDiffs.length > 0) return leftDiffs;
        let rights = trees.map(tree => tree.right);
        return diff(rights);
    }
}
```