1 Homework 2

Due Date: Mon, Mar 2; To be turned in on paper in class.
No late submissions

Important Reminder: As per the course *Academic Honesty Statement*, cheating of any kind will minimally result in your letter grade for the entire course being reduced by one level.

Please remember to justify all answers.

Note that some of the questions require you to show code. You may use a JavaScript implementation to verify your answers but you should realize that you will not have access to an implementation during exams.

You are encouraged to use the web or the library but are **required** to cite any external sources used in your answers.

1. Critique the following JavaScript code fragment and provide a better replacement: *10-points*

```
var daysOfWeek =
  ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];

function getDayOfWeek(n) {
  return daysOfWeek[n];
}
```

2. Show the implementation of a `Counter()` constructor function which maintains a counter value which is initialized to 0. It should provide two methods `val()` which returns the current counter value and `inc()` which increments and returns the current counter value.

   The operation of the counter is illustrated by the following log:

```
> c1 = new Counter()
Counter { inc: [Function], val: [Function] }
> c2 = new Counter()
Counter { inc: [Function], val: [Function] }
> c1.inc()
1
> c1.inc()
2
> c2.val()
0
> c2.inc()
1
> c2.val()
```

```
1
> c1.val()
2
>
```

It is important that your implementation does not allow any access to the counter value other than that provided by the above two functions. *5-points*

3. Novice programmer Ben Novitia asks you for help in identifying the problem in the following nodejs code:

   ```
   ...

   const DIR = `${process.env.HOME}/my-stuff`;

   function readFile(filename) {
     fs.readFile(`${DIR}/${filename}`,
       function(err, result) {
         return result;
       });
   }

   const contents = readFile('test.data');
   console.log(contents);
   ```

   What do you tell Ben? *5-points*

4. You have inherited a web site which has some code which compares shipping costs. Unfortunately this code is extremely slow. The code looks like:

   ```
   const SHIPPING_PARAMS = {
     usps: { ... },
     ...,
     ups: { ... },
     fedex: { ... },
   };

   async function cheapestShipping(fromZipCode, toZipCode,
                                   shipType)
   {
     let cheapest = 99999999;
     for (const [shipper, params] of
          Object.entries(SHIPPING_PARAMS)) {
       const cost =
         await getShipCost(params, fromZipCode, toZipCode,
   ```

```
                           shipType);
    if (cost < cheapest) cheapest = cost;
  }
  return cheapest;
}
```

Assume that `getShipCost()` is an `async` function defined elsewhere.

How would you speed up this code? Document any necessary assumptions. *15-points*

5. Given JavaScript function:

```
    function mul3(a, b, c) { return a*b*c; }
```

How would you build a function `curried_mul3` such that `curried_mul3(a)(b)(c) === mul3(a, b, c)`? *10-points*

6. Assuming no earlier variable declarations, what will be the output of the following JavaScript code when run in non-strict mode?

```
x = 11;
obj1 = { x: 21, f: function() { return this.x; } }
obj2 = { x: 31, f: function() { return this.x; } }
f = obj1.f.bind(obj2);
console.log(obj1.f() - obj2.f() + f());
```

Explain why it is so. *10-points*

7. A chain of asynchronous event handlers simply passes the result from one event to the next handler in the chain. This bears a resemblance to *continuation-passing style* where a function never returns. It simply calls a continuation function with its return value.

For example, here is a recursive function which sums up the elements of an array:

```
function sum(array, i=0) {
  if (i === array.length) {
    return 0;
  }
  else {
    return array[i] + sum(array, i+1);
  }
}
```

How would you convert this to a `sum` which never returns a value but simply passes its return value to a continuation function. Specifically, show how you would implement the function:

```
/** If sum of elements of array from elements i onwards is s,
 *  call single argument continuation function ret(s).
 */
sumCont(array, ret, i=0)
```

For example, `sumCont([1,2,3,4,5], s => console.log(s))` should output 15.

You may add additional parameters with default values. You may not use destructive assignment. *15-points*

8. Identify bugs and inadequacies in the following code:

```
/** Compute overall class average and student letter grades.
 *  Specifically, assign() is invoked with 2 arguments:
 *
 *  grades: An object which maps student names to
 *             intermediate objects.  Each intermediate object
 *             contains a property for each assignment like 'hw1'
 *             or an aggregate like 'total'.
 *  cutoffs:An object which maps the upper bound for a letter
 *             grade to the letter grade.
 *
 *  Return a 2-element array. The first element of the
 *  return'd array should be the average of the total
 *  grades for all students.  The second element of the
 *  return'd array should be an object which maps student name's
 *  to letter grades based on the student's total grade and
 *  the cutoffs.
 *
 *  Example:
 *
 *  const CUTOFFS =
 *    { 60: 'F', 65: 'D', 75: 'C', 85: 'B', '100': 'A' };
 *
 *  assign({
 *    ['Ben Bitdiddle']: { hw1: 88, ..., total: 78 },
 *    [ 'Alyssa P. Hacker']: { hw1: 100, ..., total: 97 },
 *    [ 'Louis Reasoner']: { hw1: 66, ..., total: 62 }
 *  }, CUTOFFS)
 *
 *  should result in
 *
 *  [ 79,
 *    { ['Ben Bitdiddle']: 'B',
 *       [ 'Alyssa P. Hacker']: 'A',
 *       [ 'Louis Reasoner']: 'D'
 *     }
 *   ]
 */
01 function assign(grades, cutoffs) {
02   const letters = {};
03   let c = 0
04       sum = 0;
05   for (const name in grades) {
06     var total = grades.name.total;
07     sum += total;
08     c++;
09     for (const c in cutoffs) {
10       if (total < c) letters.name = cutoffs[c];
11     }
12   }
13   return [sum/c, letters];
14 }
```

The list of bugs will be pretty objective. The inadequacies, OTOH may be more subjective. They would include

Specification Bugs Usually problems with the problem specifications or requirements are much more costly to fix than coding bugs.

Style Problems Problems with code structure, naming. These are likely to affect the cost of program maintenance.

Brittleness Slight changes in the code are likely to break other portions of the code. These may affect the cost of program evolution.

Hint: In addition to several outright bugs, the above code contains inadequacies in each of the above categories. *15-points*

9. Discuss the validity of the following statements. What is more important than whether you ultimately classify the statement as **true** or **false** is your justification for arriving at your conclusion. *15-points*

   (a) The code fragment `if (str.match(/a*|[bc]/)` has a bug.

   (b) Using a function name starting with an upper-case letter makes that function into a constructor function.

   (c) It is impossible to wrap an asynchronous function within a synchronous function.

   (d) The return value of a `then()` is always a promise.

(e) The promise returned by `Promise.all()` will become settled after a minimum time which is the sum of the times required for settlement of each of its individual argument promises.