COMP 333
Spring 2026
Final Practice Exam

The final exam is cumulative. This practice exam, **in addition to** the prior practice exams, assignment 1, in-class handouts, and exams, is intended to be a comprehensive guide for studying. This practice exam only focuses on material since the last exam. You are permitted to bring three 8.5 x 11 sheets of paper into the exam with you, as long with anything printed or handwritten on them. Both sides of both sheets can be used.

Array Routines in JavaScript

1.) Consider the following array definition in JavaScript:

```
let arr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

1.a.) Use `filter` to get an array of all even elements in `arr`.

1.b.) Use `map` to get an array of strings, where each string represents a number in `arr`. As a hint, you can call the `toString()` method on a number (e.g., `5.toString()`) in JavaScript to get its string representation.

1.c.) Use a combination of `filter` and `reduce` to get the sum of all elements in `arr` which are greater than `5`.

1.d.) Use a combination of `filter` and `reduce` to get the product of all positive elements in `arr`.

1.e.) Use `reduce` to get the last element in `arr`.

Objects in JavaScript

2.) Create an object holding a field named `"foo"`, holding the value `12`.

3.) Consider the following JavaScript code:

```
let obj1 = { 'foo': 1 };
let obj2 = { 'foo': true, 'bar': 3.14 };
console.log(obj1.foo);
console.log(obj1.bar);
console.log(obj2.foo);
console.log(obj2.bar);

let obj3 = { 'baz': 'hello', '__proto__': obj2 };
console.log(obj3.foo);
console.log(obj3.bar);
console.log(obj3.baz);

let obj4 = { 'foo': 3, '__proto__': obj1 };
console.log(obj4.foo);
console.log(obj4.bar);
console.log(obj4.baz);
```

What is the output of this code?

4.a.) Define a constructor for `Dog` objects, where each `Dog` object has a name. An example code snippet is below, illustrating usage:

```
let d = new Dog("Rover"); // line 1
console.log(d.name);      // line 2; prints Rover
```

4.b.) Define a different constructor for `Dog`, which puts a `bark` method **directly** on the `Dog` objects. The `bark` method should print `"Woof!"` when called. Example usage is below:

```
let d = new Dog("Sparky");
d.bark(); // prints Woof!
```

4.c.) Define a different constructor for `Dog`, which puts a `growl` method **directly** on the `Dog` objects. The `growl` method should print `"[dog name] growls"` when called. Example usage is below:

```
let d = new Dog("Rocky");
d.growl(); // prints Rocky growls
```

5.) Consider the following JavaScript code, where multiple methods are put onto `Pair` objects in the constructor:

```
function Pair(a, b) {
    this.first = a;
    this.second = b;
    this.getFirst = function() {
        return this.first;
    };
    this.getSecond = function() {
        return this.second;
    };
}
```

Rewrite this code so that all `Pair` objects share a single prototype object, and this prototype object contains any methods defined on `Pair`.

6.) Consider the JavaScript code below and corresponding output:

```
let three = new MyNumber(3);
let five = new MyNumber(5);

let eight = three.add(five);
let fifteen = three.multiply(five);

console.log(three.getValue());
console.log(five.getValue());
console.log(eight.getValue());
console.log(fifteen.getValue());


---OUTPUT---
3
5
8
15
```

Implement any missing code necessary to produce the above output. Multiple solutions are possible. The next page is blank in case you need it.

7.) Consider the JavaScript code and corresponding output below:

```
let obj1 = new Obj(“foo”);
console.log(obj1.field); // output: foo

let obj2 = new Obj(“bar”);
console.log(obj2.field);          // output: bar
console.log(obj2.doubleField()); // output: barbar

let obj3 = new Obj(“baz”);

console.log(obj3.field); // prints baz
```

Complete any missing elements needed to allow this code to run and produce this output.

8.) Consider the following JavaScript code:

```
function Foo() {
    this.toInt = function() {
        return 0;
    };
}

function Bar(m) {
    this.n = m;
    this.toInt = function() {
        return this.n.toInt() + 1;
    };
}

let t1 = new Foo();
console.log(t1.toInt());

let t2 = new Bar(t1);
console.log(t2.toInt());

let t3 = new Bar(new Bar(t2));
console.log(t3.toInt());
```

What is the output of this code?

9.) Consider the JavaScript code and corresponding output below:

```
let w1 = new Wrapper(1);
console.log(w1.getValue()); // prints 1

let w2 = w1.transform(e => e + 1);
console.log(w1.getValue()); // prints 1
console.log(w2.getValue()); // prints 2

let w3 = w1.transform(e => e + 7);
console.log(w1.getValue()); // prints 1
console.log(w2.getValue()); // prints 2
console.log(w3.getValue()); // prints 8
```

Complete any missing elements needed to allow this code to run and produce this output. Multiple solutions are possible. The next page is blank in case you need it.