

COMP 333 Final Practice Exam

The final exam is cumulative. This practice exam, **in addition to** the prior practice exams, assignments, 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 four 8.5 x 11 sheets of paper into the exam with you, as long as they have handwritten notes on them. Both sides of both sheets can be used. To be clear, these must be entirely handwritten.

Prototype-Based Inheritance in JavaScript

1.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
```

1.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!
```

1.c.) Define a method named growl for Dog objects, which prints "[dog name] growls" when called. Use Dog's **prototype**, instead of putting the method directly on Dog objects themselves. Example usage is below:

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

2.) Consider the JavaScript code below:

```
function Animal(name) { this.name = name; }
Animal.prototype.getName = function() { return this.name; }
function Bird(name) { this.name = name; }
Bird.prototype = { '__proto__': Animal.prototype };
Bird.prototype.fly = function() {
  console.log(this.getName() + " flies");
}
function Mouse(name) {
  this.name = name;
  this.squeak = function() {
    console.log(this.name + " squeaks");
  }
}
Mouse.prototype = { '__proto__': Animal.prototype };
Mouse.prototype.fly = Bird.prototype.fly;
let b1 = new Bird("Coco"); let b2 = new Bird("Sunny");
let m1 = new Mouse("Pip"); let m2 = new Mouse("Ruby");
```

Write a memory diagram which shows how memory looks after this program executes. Your diagram should include the objects and fields associated with b1, b2, m1, m2, Mouse, Bird, and Animal. The next page is blank if you need it.

3.) Consider the JavaScript code below, which implements immutable linked lists:

```
function List() {}
List.prototype.isList = function() { return true; }
function Cons(head, tail) {
  this.head = head;
  this.tail = tail;
}
Cons.prototype = new List();
Cons.prototype.isEmpty = function() { return false; }
function Nil() {}
Nil.prototype = new List();
Nil.prototype.isEmpty = function() { return true; }
let list1 = new Nil();
let list2 = new Cons("hi", list1);
```

Write a memory diagram which shows how memory looks after this program executes. Your diagram should include the objects and fields associated with `List`, `Cons`, `Nil`, `list1`, and `list2`. The next page is blank if you need it.

4.) 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); // output: baz
// hasOwnProperty is a built-in method which returns true if the
// object has the field directly, or false if it merely inherits
// the field.
console.log(obj3.hasOwnProperty("doubleField")); // output: false
```

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

5.) 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.

6.) Consider the JavaScript code below and corresponding output, adapted from the second assignment:

```
function Cons(head, tail) {
  this.head = head;
  this.tail = tail;
}
function Nil() {}

let list = new Cons(1, new Cons(2, new Cons(3, new Nil())));
list.forEach((x) => console.log(x));
```

---OUTPUT---

```
1
2
3
```

Implement any missing code necessary to produce the above output.

Language Concepts

7.) In 1-3 sentences, explain the difference between compilation and interpretation. Your answer does not need to be detailed enough to implement a compiler or interpreter.