COMP 333 Midterm #2 Practice Exam

This is representative of the kinds of topics and kind of questions you may be asked on the midterm. This practice exam, along with assignment 2 and the in-class handouts from list routines onwards, are intended to be comprehensive of everything on the exam. That is, I will not ask anything that's not somehow covered by those sources.

You are permitted to bring two 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.

List 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 `reduce` to get the last element in `arr`.

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

Prototype-Based Inheritance in JavaScript

2.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
```

2.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!
```

2.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
```

3.) 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.prototype`, and `Bird.prototype`, `Animal.prototype`. You do not need to show what `Animal`, `Mouse`, and `Bird` refer to.

4.) Consider the JavaScript code below, adapted from the second assignment:

```
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`.

5.) Consider the test suite below, using `assertEquals` from the second assignment:

```
function test1() {
  let t1 = new Obj("foo");
  assertEquals("foo", t1.field);
}

function test2() {
  let t2 = new Obj("bar");
  assertEquals("barbar", t2.doubleField());
}

function test3() {
  let t3 = new Obj("baz");
  // hasOwnProperty returns true if the object itself has the field,
  // otherwise it returns false.  If the field is on the object's
  // prototype instead (__proto__), it returns false.
  assertEquals(false, t3.hasOwnProperty("doubleField"));
}
```

Write JavaScript code which will make the above tests pass.

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.

7.) 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.