Name:	

Cpt S 489 – Midterm Exam – Spring 2017 (24 points total)

Guidelines/Requirements for any coding questions on this exam

- Credit is given only for working JavaScript code (no pseudo-code or written explanations).
- Unreadable/messy code is worth 0 points. Don't have stuff scribbled out or arrows redirecting lines to different places. If necessary, implement pseudo-code on back of page to first figure out the solution approach, then implement the actual JavaScript code on the problem page.
- Excessively lengthy implementations and/or implementations that involve strategies that are bad practice are worth few if any points.
- You may (and are encouraged to if it keeps your code simple and readable) create utility functions.

1. (1 point) Complete the function below such that it returns an object with properties "value", "next", and "previous", set to the corresponding parameter values.

```
function MakeLLNode(nodeValue, nodeNext, nodePrevious) {

}
```

2. (1 point) What is the output (in the console window) from calling the WhatIsTheOutput function? Recall that "console.log" writes a line (with a line break) to the console window.

Code

```
function WhatIsTheOutput() {
        var v1 = 1, v2 = 2;
        var f = function() { // Read carefully – 4 statements on next line
                var next = v1 + v2;     v1 = v2;     v2 = next;     console.log(next);
        };
        for (var i = 0; i < 3; i++) { f(); }
}
```

Output

3. (2 points) Explain the different between the Object.seal and Object.freeze functions.

4. (1 point) What does the Object.create(objParam) function call do?

5. (1 point) If Object.create is used (assume properly) to have one class inherit from another via prototype inheritance, there is still one other thing that must be done in the inheriting class constructor function. It's done automatically in languages like C++ and C#, but not JavaScript. What is it?

6. (2.5 points) Suppose an object is initialized in code with the following line:
var obj = { Value: "101" };
Complete the table by filling out the results for each statement. In other words, if each statement were assigned to a variable via "var myVar = [statement]", what would be the value of myVar? One item is given for you.

Statement	**Result / Evaluates to**
obj.Value	"101"
obj.hasOwnProperty("toString")	
obj.toString()	
obj.newProperty	
obj.x = 42;	
obj.value + "1"	

7. (3.5 points) Assume the following code exists:

```
function Animal() { }
Animal.prototype.makeSound = function() { console.log(“unknown”); }
Animal.prototype.logIsAnimal = function() { console.log(“true”); }

function Cat() { Animal.call(this); }
Cat.prototype = Object.create(Animal.prototype);
Cat.prototype.meow = Cat.prototype.makeSound = function() {
   console.log(“meow”);
}
```

Fill in the table below with the output (in the console window) that occurs as the result of the statement(s). Draw memory diagrams below the table to help you.

Statement	**Console Output**

var cat1 = new Cat();
cat1.makeSound();	

var animal1 = new Animal();
animal1.makeSound();	

var cat2 = new Cat();
cat2.logIsAnimal();	

console.log(Animal.prototype == Cat.prototype);	

var cat3 = new Cat();
Animal.prototype.meow = function()
{
  console.log(“Not all animals meow”);
};
cat3.meow();	

Animal.prototype.meow = function()
{
  console.log(“Not all animals meow”);
};
var cat4 = new Cat();
cat4.meow();	

Cat.prototype.logIsAnimal = function()
{
  console.log(“override base class”);
};
var animal2 = new Animal();
animal2.logIsAnimal();	

(can draw memory diagrams here and/or on back of page)

For problems 8 through 10, assume a red-black tree has been properly constructed and that each node has “value”, “left”, “right”, “parent”, and “red” members. The “red” property is a bool value that is true if the node is red, false if it is black. Complete the following functions that perform various operations on nodes in this tree. Assume the tree is correctly built and the rules of red-black trees hold.

```
// 8. (2 points)
// Counts the number of levels in the tree
// Worst case time complexity must be O(n)
function countLevels(node) {

}
```

```
// 9. (2 points)
// Gets the 0-based index of the specified node (root is level 0, its children level 1, and so on)
// Worst case time complexity must be O(log n)
function getLevel(node) {

}
```

```
// 10. (2 points)
// Gets the path length, in number of black nodes, from the node to its leaves. Do NOT count the null
// leaf nodes in the path length, but DO count the starting node, if it is black. Recall that the "red"
// property is used to determine if a node is red or black. (worst case time must be O(log n))
function getBlackPathLength(node) {
 // Think it through. You should not need more space than what is provided.
}
```

11. (2 points) Implement the "doesClassExist" function to return true if there's a function with the specified name (which is a string) and false otherwise. Implement with 1 line of code.

```
function doesClassExist(name) {
}
```

12. (1 point) The code below returns an array of 5 functions, each of which does an alert with a number value. The intention is to have the function alert its index. So the function at index 0 should alert "0", the function at index 1 should alert "1" and so on.

```
function make5AlertFuncs() {
   var arr = new Array();
   for (var index = 0; index < 5; index++)
      arr[index] = function() { alert(index); };
   return arr;
}
```

If it is called it will indeed return an array of functions, but calling each one would not alert its index. What does each function alert instead?

13. (1 point) Briefly explain WHY the functions from the previous problem alert what they do.

14. (2 points) Implement the fixed version of the make5AlertFuncs function, so that it actually does return functions that alert the intended value.

```
function make5AlertFuncs() {

}
```