University of Maryland College Park
Dept of Computer Science
CMSC389N Spring 2018
Midterm II Key

Last Name (PRINT): _____

First Name (PRINT): _____

University Directory ID (e.g., umcpturtle) _____

Instructions

- This exam is a closed-book and closed-notes exam.
- Total point value is 200 points.
- The exam is a 75 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY.
- **You may not use jQuery nor Bootstrap.**
- **You don't need to use meaningful variable names; however, we expect good indentation.**

Grader Use Only

#1	Problem #1 (HTML/CSS/JS Language)	(54)	
#2	Problem #2 (JavaScript Coding/Custom Types)	(56)	
#3	Problem #3 (JavaScript Coding/Dynamic HTML)	(90)	
Total	Total	(200)	

Problem #1 (HTML/CSS/JS Language)

1. (3 pts) Which of the following will always work when trying to identify an object as an array?

   a. alert
   b. instanceof
   c. Array.isArray()
   d. isNaN()
   e. None of the above

   Answer: c.

2. (3 pts) Using the => operator, initialize the variable **average** with a function that takes two parameters and returns the average.

```
let average =
```

   Answer: (x, y) => (x + y) / 2;

3. (3 pts) Complete the following template literal so the **result** variable is initialized with the string "cost is " followed by the product of **x** and **y**. You may only have one statement (just complete the template literal).

```
let x = Number(prompt("Enter Val1")), y = Number(prompt("Enter Val2"));
let result = `
```

   Answer: `cost is ${x * y}`

4. (4 pts) The function **studentInfo** has the following prototype:

```
function studentInfo(name, collegeTown, homeTown, gpa)
```

   Write a function call that will use the spread operator and the array ["Bethesda", "Wheaton"] in order to initialize the **collegeTown** and **homeTown** parameters. You can assume the **name** and **gpa** parameters are "John", and 3.7, respectively.

   Answer: `studentInfo("John", ...["Bethesda", "Wheaton"], 3.7);`

5. (3 pts) Complete the assignment to **x** so a JSON representation of the object is assigned to it.

```
var obj = { name: "Mary", age: 45 };
var x =
```

   Answer: JSON.stringify(obj);

6. (3 pts) Complete the assignment to **f1** so it refers to a **sum** function that is permanently associated with the object **obj**.

```
function sum(x, y) { return x + y + this.roomTemp; }
obj = {roomTemp: 50};

let f1 =
```

   Answer: `sum.bind(obj);`

7. (8 pts) Using the **sort** array function and anonymous functions, define the function used by **sort** that will sort the elements of the **examScores** array by decreasing **score** value.

```
let examScores = [
      { name: "Sarah", score: 99, year: 3},
      { name: "Bob", score: 50, year: 4},
      { name: "Mike", score: 78, year: 1}
];
```

```
examScores.sort(YOUR_FUNCTION_HERE)
```

One Possible Answer: `function (x, y) { return y.score - x.score; }`

8. (6 pts) Using arrow functions (=>) and **Math.random()** initialize **random_value** with a random **integer** value between 0 and 19 (includes 0 and 19).

```
let random_value =
```

Answer: `() => Math.floor(Math.random() * 20);`

9. (7 pts) Define your own **Error** type called **InvalidTemperature**. The following is an example of using your error type.

```
try {
   var value = Number(prompt("Enter positive (or 0) value"));
   if (value < 0) {
      throw new InvalidTemperature("positive value expected");
   }
} catch(error) { alert(error.message); }
```

Answer:

```
function InvalidTemperature(message) {
      this.name = "InvalidTemperature";
      this.message = message;
}
InvalidTemperature.prototype = new Error();
```

10. (14 pts) A **courses** array keeps track of courses in a college. The following is an example of some entries the array could have:

```
const courses = [ {name: "js101", type:"int", credits: 3, cost: 15.00},
                  {name: "php201", type: "adv", credits: 2, cost: 20.00},
                  {name: "c300", type: "adv", credits: 3, cost: 30.00},
                ];
```

   a. Complete the following statement so the name of each course is printed using document.writeln. Your code should work with different data (not just the entries shown above).

      Answer: `courses.forEach(i => document.writeln(i.name + "<br>"));`

   b. Complete the following statement so **threeCreditsOrMore** is initialized with an array of courses having three or more credits. Your code should work with different data (not just the entries shown above).

      Answer: `const threeCreditsOrMore = courses.filter(course => course.credits >= 3);`

   c. Complete the following statement so **costSum** is initialized with the sum of the costs of courses that have exactly three credits. For example, for the above data, **costSum** will be initialized to 45. Your code should work with different data (not just the entries shown above).
      Answer:

```
let initialValue = 0;
const costSum = courses.reduce((total, curr) => (curr.credits == 3) ? total + curr.cost: total, initialValue);
```

Problem #2 (JavaScript Coding/Custom Types)

1. (46 pts) Write **JavaScript (NOT PHP)** that defines two custom types using the approach presented in class. **If you use E6 class definitions (similar to what you have in Java) you will not receive any credit for this problem.**

    a. **Printer**
        i. Define a **Printer** custom type that has two instance variables named **maxPages** and **price** (they are not private).
        ii. Define a constructor that has two parameters: **maxPages** and **price**.
        iii. A method named **setPrice** that will update the **price** instance variable if the parameter is a number; otherwise the price will be set to 50.
        iv. Define a method **details** that returns a string with the **maxPages** and **price** (see example below for format information).
        v. **Your implementation must be efficient (i.e., do not create unnecessary objects).**

    b. **LaserPrinter**
        i. Define a **LaserPrinter** custom type that "extends" the **Printer** custom type. The type has an instance variable named **watts**; this instance variable is not private.
        ii. Define a constructor that has **maxPages**, **price**, and **watts** as parameters. The constructor will initialize the corresponding instance variables.
        iii. Define a method named **getWatts** that returns the watts.
        iv. Define a method **details** that returns a string with the **maxPages**, **price**, and **watts** (see example below for format information). This method must call the **details** method of the **Printer** custom type.
        v. **Your implementation must be efficient (i.e., do not create unnecessary objects).**

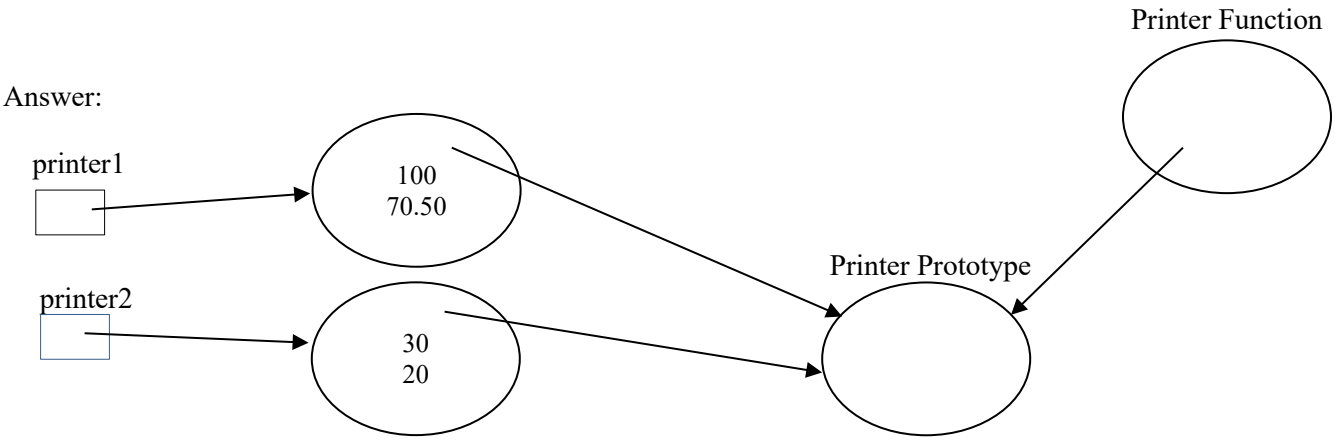
Answer

```
function Printer(maxPages, price) {
    this.maxPages = maxPages;
    this.price = price;
}

Printer.prototype.setPrice = function(price) {
    this.price = !isNaN(price) ? price : 50;
}

Printer.prototype.details = function() {
    return `MaxPages: ${this.maxPages}, Price: ${this.price}`;
}

function LaserPrinter(maxPages, price, watts) {
    Printer.call(this, maxPages, price);
    this.watts = watts;
}

LaserPrinter.prototype = new Printer();

// Ignore this one
LaserPrinter.prototype.constructor = LaserPrinter;

LaserPrinter.prototype.getWatts = function() {
    return this.watts;
}

LaserPrinter.prototype.details = function() {
    let lp = Object.getPrototypeOf(LaserPrinter.prototype);
    return lp.details.call(this) + ", Watts: " + this.watts;
}
```

2. (10 pts) Draw a diagram that illustrates the objects that are present after the following two **Printer** objects are created. Make sure you label prototype objects as such.

```
const printer1 = new Printer(100, 70.50);
const printer2 = new Printer(30, 20);
```

Answer:



Problem #3 (JavaScript Coding/Dynamic HTML)

Write a **JavaScript (NOT PHP)** program that prints a table of activities for specific days. For this problem we will only have activities for Monday, Wednesday, and Friday. The global **calendar** object (see next page) provides information about activities per day. **Although we have already defined some activities, your code must work for any activities we may place in the calendar object.** For this problem:

- Define a form with a textfield and two buttons (see example below for format information).
- Users will enter in the textfield one of the following values: Monday, Wednesday, Friday, All. In addition, multiple days can be specified by separating them by commas.
- After providing an entry and pressing the **Display Activities** button, your program will display in a table the activities for the specified day(s) (e.g., the entry ***Monday,Wednesday*** will display a table with activities for those two days). If the user provides the ***All*** entry, all the activities in the calendar will be displayed. For this problem you can assume valid data will be provided.
- The activity table will be displayed under the "Activity Calendar" heading.
- Each activity for a day will end with the | character (the symbol separates activities).
- If the user selects the **Reset Days** button, the default entry (***All***) will appear in the textfield.
- If the user clicks on the activities table (if any) the table will disappear.
- A **main** function will designate a function named **displayEntries** as the function the **DisplayActitivies** button will call when selected. It will also designate a function named **clearTable** as the function that will be called when we click on the activity table (if present). Feel free to add any other functionality to the main function you understand is needed.
- Notice that the HTML and JavaScript appears in a single file.
- Feel free to add any functions and/or custom types you understand you need.

Step1 (Form)

Day(s): All

Display Activities | Reset Days

Activity Calendar

Step2 (After Adding Entry and Selecting Display Activities)

Day(s): Monday,Wednesday

Display Activities | Reset Days

Activity Calendar

Day	Task
Monday	project|post worksheet|
Wednesday	post project|

```
let calendar = { Monday:["project", "post worksheet"],
                 Wednesday:["post project"],
                 Friday:["exam", "grading", "office hours"]}


<form>
    <strong>Day(s): </strong><input id="days" type="text" size="20" value="All"/><br><br>
    <input id="displayButton" type="button" value="Display Activities"/>
    <input type="reset" value="Reset Days"/>
</form>


<h3>Activity Calendar</h3>
<div id="displayActivities"></div>



<script>
    "use strict";

    let calendar = {
        Monday: ["project", "post worksheet"],
        Wednesday: ["post project"],
        Friday: ["exam", "grading", "office hours"]
    };

    main();

    function getElem(id) {
        return document.getElementById(id);
    }

    function main() {
        getElem("displayButton").addEventListener("click", displayEntries);
        getElem("displayActivities").addEventListener("click", clearTable);
    }

    function displayEntries() {
        let days = getElem("days").value.trim();
        let daysArray = days != "All" ? days.split(",") : ["Monday", "Wednesday", "Friday"];

        let activities = "<table border = '1'>";
        activities += "<tr><th>Day</th><th>Task</th></tr>";
        for (let day of daysArray) {
            activities += "<tr><td>" + day + "</td><td>";
            for (let task of calendar[day]) {
                activities += task + "|";
            }
            activities += "</td><tr>";
        }
         activities += "</table>";
        getElem("displayActivities").innerHTML = activities;
    }


    function clearTable() {
        getElem("displayActivities").innerHTML = "";
    }
</script>
```