



University of Maryland College Park

Dept of Computer Science

CMSC389N Fall 2017

Midterm II Key

Last Name (PRINT): _____

First Name (PRINT): _____

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

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

#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 expressions are considered true in JavaScript?

- a. `NaN == NaN`
- b. `NaN === NaN`
- c. `new Object(false)`
- d. `null`

Answer: c.

2. (3 pts) In JavaScript which value is associated with object properties that do not exist? Circle all that apply.

- a. `null`
- b. `0`
- c. `undefined`
- d. `"" /* empty string */`

Answer: c.

3. (3 pts) Rewrite the second assignment using template literals.

```
let name = "Richardson";
let introduction = "Dear Prof. " + name;
```

Answer: `let introduction = `Dear Prof. ${name}`;`

4. (4 pts) Complete the following assignment so **both** is initialized with a Set that includes the strings present in both spring and fall.

```
let spring = ["John", "Mildred"], fall = ["Rose", "Bill"];
let both = new Set(
```

One Possible Answer: `new Set([...spring, ...fall])`

Answer: (-1 pt) Per mistake; do not exceed item's total

5. (4 pts) Given the following code fragment, which of the following are VALID (circle the valid ones).

```
var car = {};
car.make = "Nelyota";
car.year = 2004;
Object.seal(car);
```

- a. `car.make = "PadYota";` **VALID**
- b. `delete car.year;`
- c. `car.year = null;` **VALID**
- d. `car.mileage = 2000;`

Answer: c. and d.

6. (4 pts) Complete the assignment to **y** so it refers to a function that is permanently associated with the object **x**. The following code will have as output 13.

```
function task(limit) { return limit + this.myConstant; }
let x = {myConstant: 10};

let y =

document.writeln(y(3));
```

Answer: `task.bind(x);`

7. (6 pts) Write code that will display the contents of a map by printing (using `document.writeln`) the key, a comma and the value. For example, for the map below the output generated will be:

```
195, 60
270, 8

let map = new Map();
map.set("195", 60);
map.set("270", 8);
```

One Possible Answer:

```
for (let [key, value] of map) {
  document.writeln(key + ", " + value + "<br>");
}
```

8. (6 pts) Using arrow functions (`=>`), initialize **divisible** with a function that returns the string “No” if the first value is not divisible by the second, and “Yes” otherwise. For example, `divisible(4, 2)` will return “Yes”; `divisible(4, 3)` will return “No”.

```
let divisible =
```

One Possible Answer: `(x, y) => x % y !== 0 ? "No" : "Yes";`

9. (7 pts) Complete the following assignment so the variable **obj** is initialized with an object whose JSON representation is stored in `localStorage` under the name “data”.

```
let obj =
```

One Possible Answer:

```
JSON.parse(localStorage.getItem("data"));
```

10. (14 pts) A **restaurants** array keeps track of visits a person has done to several restaurants. The following is an example of some entries the array could have:

```
const restaurants = [{name: "BurgerPlace", type: "fast", stars: 3, cost: 8.00},
                     {name: "HomeFood", type: "slow", stars: 5, cost: 20.00},
                     {name: "PizzaPlace", type: "fast", stars: 2, cost: 7.50},
                     ];
```

- a. (3 pts) Complete the following statement so the name of each restaurant is printed using `document.writeln`. Your code should work with different data (not just the entries shown above).

```
restaurants.forEach(
```

Answer:

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

- b. (4 pts) Complete the following statement so **threeStarsOrMore** is initialized with an array of objects having restaurants with three or more stars. Your code should work with different data (not just the entries shown above).

```
const threeStarsOrMore = restaurants.filter(
```

Answer:

```
const threeStarsOrMore = restaurants.filter(rest => rest.stars >= 3);
```

- c. (7 pts) Complete the following statement so **costSum** is initialized with the sum of costs that are greater than or equal to 8. For example, for the above data, **costSum** will be initialized to 28. Your code should work with different data (not just the entries shown above).

```
const costSum = restaurants.reduce(
```

Answer:

```
const costSum = restaurants.reduce((accumulated, curr) => (curr.cost >= 8) ?  
accumulated + curr.cost: accumulated, 0);
```

Problem #2 (JavaScript Coding/Custom Types)

1. (46 pts) Write **JavaScript (NOT PHP)** that defines two custom types (“classes”) 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. **Photo**
 - i. Define a **Photo** custom type that has two instance variables named **description** and **date** (they are not private).
 - ii. Define a constructor that has two parameters: description and date.
 - iii. A method named **setDescription** that will update the description instance variable if the parameter is not the empty string or a string with empty spaces.
 - iv. Define a method **info** that returns a string with the description and date (see example below for format information).
 - v. **Your “class” must be efficient; that means you should not create unnecessary objects.**
 - b. **DigitalPhoto**
 - i. Define a **DigitalPhoto** custom type that “extends” the **Photo** custom type. The class has an instance variable named **size**; this instance variable is not private.
 - ii. Define a constructor that has description, date, and size as parameters. The constructor will initialize the corresponding instance variables.
 - iii. Define a method named **getSize** that returns the size.
 - iv. Define a method **info** that returns a string with the description, date, and size (see example below for format information). This method must call the **info** method of the **Photo** custom type.
 - v. **Your “class” must be efficient; that means you should not create unnecessary objects.**
2. (10 pts) Draw a diagram that illustrates the objects that are present after the following two Photo objects are created. Make sure you label prototype objects as such.

```
const photo1 = new Photo("A", "today");  
const photo2 = new Photo("B", "tomorrow");
```

The following is an example of using the custom types you need to define.

<pre>const photo = new Photo("Stamp Union", "12/01/17"); document.writeln(photo.info() + "
"); photo.setDescription("Gym"); document.writeln(photo.info() + "
"); document.writeln("
*** Digital Photo ***
"); const dphoto = new DigitalPhoto("CSIC", "11/18/17", 2); document.writeln(dphoto.info() + "
"); document.writeln("Size: " + dphoto.getSize() + "
"); dphoto.setDescription(" "); document.writeln("After using invalid description
"); document.writeln(dphoto.info() + "
");</pre>	Output Description: Stamp Union, Date: 12/01/17 Description: Gym, Date: 12/01/17 *** Digital Photo *** Description: CSIC, Date: 11/18/17, Size: 2 Size: 2 After using invalid description Description: CSIC, Date: 11/18/17, Size: 2
--	--

Answer (46 pts):

(4 pts) Constructor

```
function Photo(description, date) {
    this.description = description;
    this.date = date;
}
```

(8 pts) setDescription

```
Photo.prototype.setDescription = function(description) {
    if (description.trim() != "") {
        this.description = description;
    }
}
```

(8 pts) info

```
Photo.prototype.info = function() {
    return `Description: ${this.description}, Date: ${this.date}`;
}
```

(8 pts) Constructor (DigitalPhoto)

```
function DigitalPhoto(description, date, size) {
    Photo.call(this, description, date);
    this.size = size;
}
```

(4 pts) Prototype

```
DigitalPhoto.prototype = new Photo();
```

```
// Ignore this one
```

```
DigitalPhoto.prototype.constructor = DigitalPhoto;
```

(4 pts) getSize

```
DigitalPhoto.prototype.getSize = function() {
    return this.size;
}
```

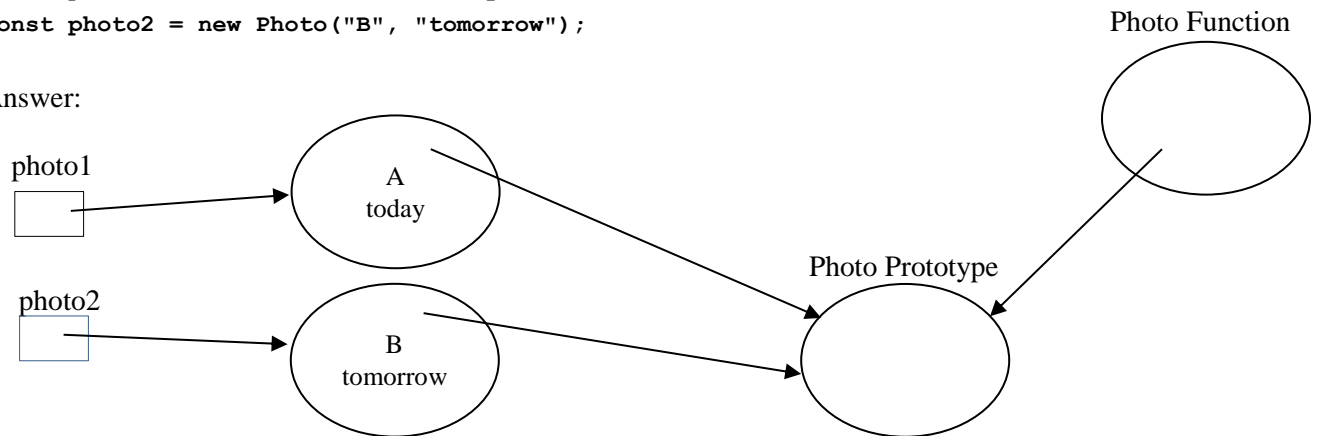
(10 pts) info (DigitalInfo)

```
DigitalPhoto.prototype.info = function() {
    let dp = Object.getPrototypeOf(DigitalPhoto.prototype);
    return dp.info.call(this) + ", Size: " + this.size;
}
```

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

```
const photo1 = new Photo("A", "today");
const photo2 = new Photo("B", "tomorrow");
```

Answer:



Problem #3 (JavaScript Coding/Dynamic HTML)

Write a **JavaScript (NOT PHP)** program that allow us to keep track of assistance students receive in a TA room. For this problem:

- Define a form with a textfield and two buttons (see example below for format information).
- Users will enter in the textfield an entry that has the following format:

<DIRECTORY_ID>,<DESCRIPTION>,<DURATION>

The string provides the student's directory id, followed by a description of the assistance the student received, followed by the amount of time the TA provided assistance (e.g., **terp, debugging code, 20**). **Each of these values is separated by a comma.**

- After providing an entry and pressing the **AddEntry** button, your program will add the entry under the section labeled "Entries". If the entry provided is empty or if not all values (directory id, etc.) have been provided, the program will display the message "Invalid entry" (using alert) and will perform no further processing. You can assume that if the entry has two commas, all the fields have been provided.
- If the user selects the **Summary** button, your program will display under the section labeled "Summary" the number of entries that have been submitted followed by the entries **sorted by duration**. No commas will separate the components of an entry.
- A **main** function will designate a function named **addEntry** as the function the **AddEntry** button will call when selected. It will also designate a function named **summary** as the function that will be called when the **Summary** button is selected. 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.

Form (Step1)

Entry:

AddEntry

Summary

Entries

Summary

After Adding Three Entries (Step2)

Entry:

AddEntry

Summary

Entries

mary, debugging code, 15
terps, testing, 7
peter, validation, 9

Summary

After Selecting Summary (Step3)

Entry:

AddEntry

Summary

Entries

mary, debugging code, 15
terps, testing, 7
peter, validation, 9

Summary

Total: 3
terps testing 7
peter validation 9
mary debugging code 15

Answer (Form/HTML) (90 pts):

```
<strong>Entry</strong>: <input id="entry" type="text" /><br><br>
                        <input id="addEntryButton" type="button" value="AddEntry" />
                        <input id="summaryButton" type="button" value="Summary" />
<div id="displayEntries">
  <h3>Entries</h2>
</div>
<div id="displaySummary">
  <h3>Summary</h2>
</div>

<script>
  "use strict";

  let allEntries = new Array();

  // auxiliary function students don't need to implement
  function getElem(id) {
    return document.getElementById(id);
  }

  function main() {
    getElem("addEntryButton").addEventListener("click", addEntry);
    getElem("summaryButton").addEventListener("click", summary);
  }
</script>
```

```

function addEntry() {
    let entry = getElem("entry").value;
    let components = entry.split(",");
    if (components.length != 3) {
        alert("Invalid entry");
    } else {
        getElem("displayEntries").innerHTML += entry + "<br>";
        let studentEntry = {};
        studentEntry.id = components[0].trim();
        studentEntry.description = components[1].trim();
        studentEntry.duration = Number(components[2]);
        allEntries.push(studentEntry);
    }
}

function summary() {
    getElem("displaySummary").innerHTML = "<h3>Summary</h2>Total: " +
allEntries.length + "<br>";
    allEntries.sort(function(x, y) { return x.duration - y.duration; });
    allEntries.forEach(function(elem) {
        getElem("displaySummary").innerHTML += (elem.id + " " +
elem.description + " " + elem.duration + "<br>");
    });
}
</script>

```