



University of Maryland College Park

Dept of Computer Science

CMSC389N Summer 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: _____

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)	(60)	
#2	Problem #2 (JavaScript Coding/Custom Types)	(55)	
#3	Problem #3 (JavaScript Coding/Dynamic HTML)	(85)	
Total	Total	(200)	

Problem #1 (HTML/CSS/JS Language)

1. (3 pts) Write a single JavaScript statement that will remove the last two elements from the following array.

```
const languages = ["Up", "Down", "Center"];
```

Answer: `languages.length = 1`

2. (3 pts) What is the output of the following code fragment? If the code will not execute and generate an error in the console, write "ERROR".

```
const cities = ["SilverS", "CollegeP", "Bethesda"];
cities[5] = "Bethesda";
document.writeln(cities[3] + " " + cities[5] + "<br>");
```

Answer: undefined Bethesda

3. (3 pts) Complete the assignment to the **mult** variable so it becomes a function that has as parameter and body the values specified in the **param** and **body** variables, respectively.

```
const param = prompt("Enter parameter");
const body = prompt("Enter function body");
const mult =
```

Answer: `new Function(param, body);`

4. (3 pts) Is there a difference between `Number.isNaN` and `isNaN`? If there is no difference write NONE; otherwise specify the difference.

Answer: Yes, they are different. `isNaN` tries to convert argument into a number.

5. (3 pts) In a weakmap, what takes place when an object that is a key in the map only has one reference (the one associated with the map)?

Answer: The object is garbage collected.

6. (4 pts) Complete the left side of the second assignment so variables `x` and `y` are initialized with the first two array entries. The output of the code fragment below is: *x: Fortran, y: Java*

```
let languages = ["Fortran", "Java", "Python"];

let _____ = languages;

document.writeln("x: " + x + ", y: " + y + "<br>");
```

Answer: `[x, y]`

7. (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);`

8. (6 pts) Provide an expression that relies on `Math.random()` that assigns to `x` a random integer value in the range [1, 6]. Notice the range includes 1 and 6.

```
let x =
```

Answer: `Math.floor(6 * Math.random()) + 1`

9. (8 pts) Write code that will compute the sum of the values of entries of the map where the key does not start with letter "B". For example, the sum for the below map will be 100.

```
let map = new Map();
map.set("Bethesda", 200).set("CP", 60).set("Olney", 40);
```

Possible Answer:

```
let s = 0;
for (let [key, val] of map) {
  if (key[0] != "B") {
    s += val;
  }
}
```

10. (8 pts) Using the `sort` array function and arrow functions (`=>`), define the function used by `sort` that will sort the elements of the **employee** array by name (you can sort in increasing or decreasing alphabetical order).

```
let employees = [
  { name: "Bob", gpa: 2.7},
  { name: "Albert", gpa: 3.0},
  { name: "Linda", gpa: 3.3}
];
```

Answer: `employees.sort((x, y) => { return x.name <= y.name });`

11. (8 pts) Define a function called **product** that will compute the product of the argument values. The function can take 1 or more arguments. For example, `product(2)` will return 2, `product(3, 5)` will return 15, `product(2, 3, 4)` will return 24. The number of arguments is not limited to 3; could be any number of arguments.

One Possible Answer:

```
function product(...values) {
  let answer = 1;
  for (let val of values) {
    answer *= val;
  }
  return answer;
}
```

12. (7 pts) Define your own Error type called **InvalidPasswordError**. The following is an example of the using your Error type.

```
try {
  let code = prompt("Enter password");
  if (code != "terps")
    throw new InvalidPasswordError("Invalid password");
  document.writeln("Welcome<br>");
} catch(error) {
  alert(error.message);
}
```

Answer:

```
function InvalidPasswordError(message) {  
    this.message = message;  
}  
InvalidPasswordError.prototype = new Error();
```

Problem #2 (JavaScript Coding/Custom Types)

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.**

1. Book

- a. Define a **Book** custom type that has two instance variables named **title** and **price**. The only instance variable that is private is **title**.
- b. Define a constructor that has two parameters: title and price.
- c. Instances of the “class” will have access to a method named **getTitle** that returns the title.
- d. Instance of the “class will have access to a method named **setPrice** that will change the price instance variable if the parameter is greater than or equal to zero; otherwise no change will take place.
- e. Define a method **print** that returns a string with the title and price (see example below for format information).
- f. **Your “class” must be efficient; that means you should not create unnecessary objects.**

2. ElectronicBook

- a. Define an **ElectronicBook** custom type that “extends” the **Book** custom type. The class has an instance variable named **megabytes**; this instance variable is not private.
- b. Define a constructor that has title, price, and megabytes as parameters. The constructor will initialize the corresponding instance variables.
- c. Define a method named **getMegabytes** that returns the number of megabytes.
- d. Define a method **print** that return a string with the title, price, and megabytes (see example below for format information). This method must call the print method of the **Book** custom type.
- e. **Your “class” must be efficient; that means you should not create unnecessary objects.**

Book

Answer:

```
function Book(title, priceIn) {
    this.price = priceIn;

    this.getTitle = function() {
        return title;
    }
}

Book.prototype.setPrice = function(priceIn) {
    if (priceIn >= 0) {
        this.price = priceIn;
    }
}

Book.prototype.print = function() {
    return `Title: ${this.getTitle()}, Price: ${this.price}`;
}
```

ElectronicBook

Answer:

```
function ElectronicBook(title, price, megabytes) {
    Book.call(this, title, price); this.megabytes = megabytes;
}

ElectronicBook.prototype = new Book();

ElectronicBook.prototype.constructor = ElectronicBook;

ElectronicBook.prototype.getMegabytes = function() {
    return this.megabytes;
}

ElectronicBook.prototype.print = function() {
    let p = Object.getPrototypeOf(ElectronicBook.prototype); return
    p.print.call(this) + ", Megabytes: " + this.megabytes;
}
```

Problem #3 (JavaScript Coding/Dynamic HTML)

Write a **JavaScript (NOT PHP)** program that allow us to display an unordered list of square roots based on a range provided via a textfield. For this problem:

- Define a form with a textfield and two buttons (see example below for format information).
- Users will enter in the textfield a string of the form <START_RANGE>,<END_RANGE> (e.g., 1, 3) that represents the range of square roots to display; notice there is a comma between the numbers.
- The textfield has as default value 1,3.
- Two tasks will take place when the user selects the “ShowList” button:
 - a. An unordered list of square roots starting at <START_RANGE> and ending at <END_RANGE> will be displayed after the buttons; the header “Square Roots” will precede the values. You can use the function `Math.sqrt()` to compute the square root.
 - b. The list that was generated will be stored using `localStorage`.
- When the user selects the “Load” button, the list stored in `localStorage` will be displayed after the buttons. If there is no list stored, the message “Nothing stored” will be displayed after the buttons.
- Your code must perform the following validation on the value provided in the textfield:
 - a. The message “Error: missing values” must be printed below the buttons if the value provided is an empty string or a string with only spaces.
 - b. The message “Error: Number(s) required” must be printed below the buttons if either (or both) of the values the comma separates is not a number (e.g., **bla,3** OR **3, bla**) or either or both are missing (e.g., **3,** OR **,4** OR **,**).
 - c. You can assume the string provided by the user has a single comma if the string is not the empty string or a string with empty spaces.
- A **main** function will define a function named **showList** as the function the **ShowList** button will call when 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 in addition to the **showList** and **main** functions.

Form

Range:

After Selecting ShowList Button

Range:

Square Roots

- 1(1)
- 2(1.4142135623730951)
- 3(1.7320508075688772)

After Providing Empty String and Selecting ShowList Button

Range:

Error: values missing

After Providing Invalid Range and Selecting ShowList Button

Range:

Error: Number(s) required

One Possible Answer:

```
<body onload="main()">
  Range: <input id="limit" type="text" value="1,3" size="3" />
        <input id="button" type="button" value="ShowList" />
        <input id="loadButton" type="button" value="Load" />
  <div id="display"></div>

  <script>
    function main() {
      document.getElementById("button").addEventListener("click", showList);
      document.getElementById("button").addEventListener("click", save);
      document.getElementById("loadButton").addEventListener("click", load);
    }

    function display(contents) {
      document.getElementById("display").innerHTML = contents;
    }

    function showList() {
      let limit = document.getElementById("limit").value;
      if (limit.trim() == "") {
        display("Error: values missing");
      } else {
        let vals = limit.split(",");

        if (vals[0].trim() == "" || vals[1].trim() == "" ||
            isNaN(vals[0]) || isNaN(vals[1])) {
          display("Error: Number(s) required");
        } else {
          let list = "<h2>Square Roots</h2><ul>";

          for (let i = Number(vals[0]); i <= Number(vals[1]); i++) {
            list += `<li>${i} (${Math.sqrt(i)})</li>`;
          }
          list += "</ul>";
          display(list);
        }
      }
    }

    function save() {
      localStorage.setItem("list", document.getElementById("display").innerHTML);
    }

    function load() {
      const list = localStorage.getItem("list");
      if (list == null) {
        display("Nothing stored");
      } else {
        display(list);
      }
    }
  </script>
</body>
```