University of Maryland College Park
Department of Computer Science
CMSC389N Spring 2018
Final Exam 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 120 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY.
- **You don't need to use meaningful variable names; however, we expect good indentation.**

Grader Use Only

#1	Problem #1 (Miscellaneous)	(75)	
#2	Problem #2 (Javascript Custom Types)	(35)	
#3	Problem #3 (JavaScript Coding)	(45)	
#4	Problem #4 (PHP Coding)	(45)	
Total	Total	(200)	

Problem #1 (Miscellaneous)

1. (3 pts) Define a PHP associative array ($my_words) that will map the following English words to the corresponding Spanish words:

   a. House → Casa
   b. Friend → Amigo

   $my_words =

   One Possible Answer (check any valid ones): $my_words = array("House" => "Casa", "Friend" => "Amigo");

2. (3 pts) Using a PHP foreach construct display the key and value associated with each entry of the $my_words array you defined above.

   One Possible Answer (check any valid ones): foreach ($my_words as $key => $value)

3. (3 pts) Use the php **header** function to visit the UMD web site http://www.umd.edu.

   Answer: Location: header(http://www.umd.edu)

4. (3 pts) Where is session information (i.e., key/value pairs) stored?

   a. In a file in the server (one file per session).
   b. In a single file in the server (all sessions share this file).
   c. In a cookie.
   d. In several cookies (one per key/value pair).
   e. None of the above.

   Answer: a.

5. (3 pts) The following function is expected to print the value of the global variable $myGlobal. Do you see any problems in the code? Yes or No.

```
$myGlobal = 200;

function scopeCheckThree() {
      print("In scopeCheckThree: ".$myGlobal."<br />");
}
```

   Answer: Yes (missing global $myGlobal in the function).

6. (3 pts) In PHP methods that start with two underscores are called:

   a. selected methods
   b. magic methods
   c. inherited methods
   d. implicit methods
   e. None of the above.

   Answer: b.

7. (3 pts) Write an SQL command that will delete all the records from a table called **purchases** that have an **id** field value greater than or equal to 400.

   Answer: delete from purchases where id >= 400;

8. (3 pts) Using the PHP array function **explode** complete the following code so that $result is initialized with an array that has just the numbers (without the |).

```
$data = "10|20|30|40|50|60";

$result =
```

Answer: explode("|", $data)

9. (3 pts) The body of a JavaScript function is the single variable declaration **var x;** followed by a statement that prints the value. What is the value printed?

    a. 0
    b. A trash value
    c. null
    d. undefined

Answer: d.

10. (3 pts) In JavaScript which value is associated with object properties that do not exist?

    a. undefined
    b. null
    c. a. and b.
    d. None of the above.

Answer: a

11. (3 pts) Rewrite the following JavaScript code without using [ ] assuming myData is an object.

```
myData["mary"] = 20;
```

Answer: myData.mary = 20;

12. (3 pts) Complete the following localStorage statement so we can associate with “name” the value “Mary”.

    localStorage.

Answer: localStorage.setItem("name", "Mary");

13. (3 pts) What is the output of the following JavaScript code fragment?

```
let y = 300;

if (y > 5) {
     let y = 88;
} else {
     let y = -7;
}
document.writeln("F: " + y + "<br>")
```

Answer: F: 300

14. (3 pts) The JavaScript function **init** has the following prototype:

```
function init(name, gpa, city)
```

Write a function call that will use the spread operator and the array ["Mike", 3.7] in order to initialize the **name** and **gpa** parameters. You can assume the city parameter is "Greenbelt".

Answer: init(...["Mike", 3.7], "Greenbelt")

15. (3 pts) Complete the JavaScript 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);

16. (3 pts) Complete the assignment below (using JavaScript) so the reversed string ("Bob*Tom*Al") is assigned to **result.** You may only use a single assignment otherwise you will not receive credit.

```
let line = "Al*Tom*Bob";

let result =
```

Answer: `line.split("*").reverse().join("*")`

17. (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.

18. (3 pts) Node processes the queue events in random order, invoking the appropriate callback function with information about the event. **True / False**.

Answer: False

19. (3 pts) Which of the following is an architectural style used to develop web services?

    a. REST
    b. JSON
    c. Node
    d. None of the above.

Answer: REST

20. (3 pts) In **Express** middleware is a

    a. Function
    b. Database system
    c. Variable
    d. None of the above.

    Answer: a.

21. (3 pts) **Express** is an abstraction layer on top of Node's http-server. **True / False**

    Answer: True

22. (3 pts) **Express** expands the **request** and **response** objects. **True / False**

    Answer: True

23. (9 pts) The following array will be used for the questions that follow.

```
const tas = [
                {name: "John", course: "cmsc131", hours: 4},
                {name: "Mary", course: "math106", hours: 10},
                {name: "Laura", course: "psyc101", hours: 3},
                {name: "Kelly", course: "cmsc131", hours: 5},
            ];
```

    a. Using the array **map** method initialize the **allCourses** variable with an array that has the courses each TA is associated with. Only one assignment can be used.

    ```
    const allCourses =
    ```

    Answer: tas.map(ta => ta.course)

    b. Using the array **filter** method initialize the **allcmsc131Tas** variable with an array that has TAs associated with cmsc131. Only one assignment can be used.

    ```
    const allcmsc131Tas =
    ```

    Answer: tas.filter(ta => ta.course === "cmsc131")

    c. Using the array **findIndex** method initialize the **laurasIndex** variable with the index associated with the entry that corresponds to Laura. Only one assignment can be used.

    ```
    const laurasIndex =
    ```

    Answer: tas.findIndex(ta => ta.name === "Laura")

Problem #2 (JavaScript 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. Your custom types must be efficient; that means you should not create unnecessary objects.

1. **Dessert**
    a. Define a **Dessert** custom type that has one instance variable named **name**. The instance variable is **NOT** private.
    b. Define a constructor that has as parameter the name of the dessert.
    c. Define a method **info** that returns a string with the name of the dessert (see example below for format information).

2. **VegiDessert**
    a. Define a **VegiDessert** custom type that "extends" the **Dessert** custom type. The type has an instance variable named **vegetable**; this instance variable is **NOT** private.
    b. Define a constructor that has name and vegetable as parameters. The constructor will initialize the corresponding instance variables.
    c. Define a method named **getVegetable** that returns the vegetable value.
    d. Define a method **info** that returns a string with the name and vegetable (see example below for format information). This method must call the **info** method of the **Dessert** custom type.

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

Code

```
const dessert = new Dessert("IceCream");
document.writeln(dessert.info() + "<br>");

const vegiDessert = new VegiDessert("Surprise", "Lettuce");
document.writeln("Vegetable: " + vegiDessert.getVegetable() + "<br>");
document.writeln("VegiDessert: " + vegiDessert.info());
```

Output

```
Name: IceCream
Vegetable: Lettuce
VegiDessert: Name: Surprise, Vegetable: Lettuce
```

Dessert - Answer

```
function Dessert(name) {
      this.name = name;
}


Dessert.prototype.info = function() {
      return `Name: ${this.name}`;
}
```

VegiDessert - Answer:

```
function VegiDessert(name, vegetable) {
    Dessert.call(this, name);
    this.vegetable = vegetable;
}


VegiDessert.prototype = new Dessert();


VegiDessert.prototype.constructor = VegiDessert;


VegiDessert.prototype.getVegetable = function() {
     return this.vegetable;
}

VegiDessert.prototype.info = function() {
     let p = Object.getPrototypeOf(VegiDessert.prototype);
     return p.info.call(this) + ", Vegetable: " + this.vegetable;
 }
```

Problem #3 (JavaScript Coding)

For this problem you will implement three **JavaScript (NOT PHP)** functions that support a system (see next page) that prints receipts for a supermarket. Users will provide information about an item in a textfield using the format <ITEM>,<ITEMS_PRICE>. For example, ***milk, 1.50*** represents the item milk with a price of $1.50. For this problem:

- The **addItem** function will process the entry provided in the textfield. It will remove any blank spaces surrounding the entry and it will determine whether it is valid. A valid entry is one where two values are separated by a comma. You can assume that if two values are present, the first represents the item's name and the second the price. If a valid entry is provided, the function will display in the element with id "display" the word "Added" followed by the item's name, followed by "(item #<ITEM_NUMBER>)". For example, the first item entered will have 1 as item number. If an invalid entry is provided, the message "Invalid entry" will appear in the in the element with id "display".
- The **displayReceipt** function will generate a table with all the items entered so far. Each table row will have the item's name followed by the price. The last row will display the total associated with the items (see sample below for format information). The table is displayed in the element with id "display".
- The **clear** function clears the contents of the element with id "display" and the textfield. Any information about items entered so far will be removed. After executing clear we can start a new receipt.
- Feel free to add any global variables you need, however, you may not add any additional functions.
- Keep in mind that the same item (e.g., ***milk, 1.50***) can be added multiple times.
- Feel free to use the **getElem** function we have provided.

Step 1

Item, Price:

Add Item | Display Receipt | Clear

Display

Step 2 (Entering milk item and selecting Add Item)

Item, Price: milk, 1.5

Add Item | Display Receipt | Clear

Display

Added milk(item #1)

Step 3 (Entering newspaper item and selecting Add Item)

Item, Price: newspaper, 2.0

Add Item | Display Receipt | Clear

Display

Added newspaper(item #2)

Step 4 (Selecting Display Receipt)

Item, Price: newspaper, 2.0

Add Item | Display Receipt | Clear

Display

milk	1.5
newspaper	2.0
Total:	$3.5

Step 5 (Selecting Clear)

Item, Price:

Add Item | Display Receipt | Clear

Display

PAGE FOR YOUR CODE

```
<form>
    <strong>Item, Price: </strong><input id="item" type="text" size="20"/><br><br>
    <input id="addButton" type="button" value="Add Item"/>
    <input id="displayReceiptButton" type="button" value="Display Receipt"/>
    <input id="clear" type="button" value="Clear"/>
</form>

<h3>Display</h3>
<div id="display"></div>

<script>
    "use strict";
    main();

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

    function main() {
        getElem("addButton").addEventListener("click", addItem);
        getElem("displayReceiptButton").addEventListener("click", displayReceipt);
        getElem("clear").addEventListener("click", clear);
    }
```

Answer

```
let allItems = new Array();
let currentItem = 0;

main();

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

function main() {
    getElem("addButton").addEventListener("click", addItem);
    getElem("displayReceiptButton").addEventListener("click", displayReceipt);
    getElem("clear").addEventListener("click", clear);
}


function addItem() {
    let item = getElem("item").value.trim();
    let itemArray = item.split(","), message;

    if (itemArray.length != 2) {
        message = "Invalid entry";
    } else {
        allItems[currentItem++] = itemArray;
        message = "Added " + itemArray[0] + "(item #" + currentItem + ")";
    }
    getElem("display").innerHTML = message;
}


function displayReceipt() {
    let receipt = "<table border='1'>";
    let total = 0;

    for (let i in allItems) {
        receipt += "<tr><td>" + allItems[i][0] + "</td>";
        receipt += "<td>" + allItems[i][1] + "</td></tr>";
        total += Number(allItems[i][1]);
    }
    receipt += "<tr><td>Total: </td><td>$" + total + "</td></tr>"
    receipt += "</table>";

    getElem("display").innerHTML = receipt;
}

function clear() {
    getElem("display").innerHTML = "";
    allItems = new Array();
    currentItem = 0;
    getElem("item").value = "";
}
```

Problem #4 (PHP Coding)

For this problem you will implement a **PHP application** that determines whether classes will take place after a snow storm. The program consists of three files that you need to implement.

1. **request.html** → Displays a form where the user will provide a snow amount (see image below). The form will use post and will call the script **verify.php**.

Answer:

```
<form action="verify.php" method="post">
     Snow Amount: <input type="text" name="amount" />
    <input type="submit" value="Enter value and click to find out if classes cancelled" />
</form>
```

2. **verify.php** → This script will determine whether classes will be cancelled or not. Classes will be cancelled if the amount of snow is more than 10 inches; otherwise classes will not be cancelled. **Using sessions** this script will pass the number of inches and whether classes will take place or not to the **confirmation.php** script. The method used is post as well. The following image illustrates what **verify.php** should show after the number of inches of snow have been evaluated:

Answer:

```
<?php
       require_once("support.php");

       session_start();
       $amount = $_POST['amount'];
       $result = ($amount <= 10) ? "NOT_CANCELLED" : "CANCELLED";

       $_SESSION['amount'] = $amount;
       $_SESSION['result'] = $result;

       $body = <<<EOFDATA
              <form action="confirmation.php" method="post">
                     Classes status evaluated. <input type="submit" value="Click on button to know
results" />
              </form>
EOFDATA;

echo generatePage($body);
?>
```

3. **confirmation.php** → **Using sessions** this script will retrieve the inches and decision regarding classes. If classes are not cancelled the script will display a page with the message:

"With <AMOUNT_INCHES> inches classes NOT_CANCELLED"

where <AMOUNT_INCHES> corresponds to the snow amount

Otherwise the message to print will be:

"With <AMOUNT_INCHES> inches classes CANCELLED"

For this problem feel free to use the function generatePage() we saw in class that allows you to generate an HTML document when you provide the body (e.g., generatePage($body)). Assume this function is in the file support.php (make sure you include it). **You may not use JavaScript for this problem.**

Answer:

```
<?php
require_once("support.php");

session_start();
$body = "With {$_SESSION['amount']} inches classes {$_SESSION['result']}";

echo generatePage($body);
?>
```