



University of Maryland College Park

Dept of Computer Science

CMSC389N Fall 2017

Midterm I

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 any JavaScript.**
- **You don't need to use meaningful variable names; however, we expect good indentation.**

Grader Use Only

#1	Problem #1 (Miscellaneous)	(20)	
#2	Problem #2 (PHP Coding I)	(30)	
#3	Problem #3 (PHP Coding II)	(60)	
#4	Problem #4 (PHP Coding III)	(90)	
Total	Total	(200)	

Problem #1 (HTML/CSS Concepts)

1. (3 pts) Write a SQL command that will create a table named “appts” that has the fields **student** (string 14 characters), **advisor** (string 3 characters), and **room** (integer).

2. (3 pts) Write a SQL command that will insert a record into the “appts” table above for a student named “Kelly”, where the advisor is “JSR”, and the room number is 5.

3. (3 pts) Write a SQL command that will display the name of students that are associated with the advisor named “JSR”.

4. (3 pts) Which of the following are considered false in PHP? Circle all that apply.
 - a. Empty string
 - b. String with empty spaces.
 - c. 0
 - d. Undefined variable
 - e. None of the above.

5. (3 pts) Which of the following unsets a variable? Circle all that apply.
 - a. Assigning null to the variable.
 - b. Assigning 0 to the variable.
 - c. Using unset on the variable.
 - d. Assigning the empty string to the variable.

6. (3 pts) Which of the following variables are considered empty? Circle all that apply.
 - a. \$a = 0.0;
 - b. \$b = 0;
 - c. \$c = "false";
 - d. \$d = "";

7. (2 pts) Define a PHP constant called **PRESSURE** that has as value 109.56.

Problem #2 (PHP Coding)

Define two PHP classes called **Door** and **ElectricDoor**. The specification for these classes is provided below. Use **type hints** for parameters and return values. You can assume the declaration `declare(strict_types=1)` already exists.

1. **Door** class

- a. The class has three private instance variables called **\$make**, **\$width**, and **\$height**. The first variable represents a string and the last two represent floating-point values.
- b. A static variable named **\$totalDoors** keeps track of the number of **Door** objects created.
- c. A constructor that initializes a **Door** object; it has make, width, and height as parameters.
- d. A **toString** method that returns a string with the door's area. See the output below for format information.
- e. **getMake** – get method for \$make field.
- f. **getTotalDoors** – static method that returns total number of **Doors** objects created.

2. **ElectricDoor** class

- a. The class extends the **Door** class and has a private instance variable called **\$wattage**.
- b. A constructor that has \$make, \$width, \$height and \$wattage as parameters.
- c. A **toString** method that returns a string with the door's area and the wattage. See the output below for format information.

Below we have provided a main() function (and expected output) that relies on the classes you are expected to write.

main function

```
function main() {  
    $door1 = new Door("DoorStore", 2, 7.5);  
    echo $door1."<br>";  
  
    $door2 = new ElectricDoor("EDoorSrc", 2.5, 6, 220);  
    echo $door2."<br>";  
}
```

Output

```
Area: 15  
Area: 15, Wattage: 220  
Total Doors: 2
```

PAGE FOR YOUR CODE

PAGE FOR YOUR CODE

PAGE FOR YOUR CODE

Problem #3 (PHP Coding)

Write a PHP script that generates a horizontal histogram based on numbers found in a file. The specifications for this problem are:

1. Define a text field (with default value **data.txt**) that allow us to enter a file name.
2. Two submit buttons named “PrintHistogram” and “ClearHistogram” will allow us to generate a histogram or remove it.
3. When the user selects the **PrintHistogram** button, the file specified in the file text field will be opened and a horizontal histogram will be generated based on the numbers read.
4. To generate the histogram, define a table row where the number of cells corresponds to the number read for a histogram row. Use two spaces (use) for each cell. The background of a row will be either red (odd-numbered row) or blue (even-numbered row). The table should have its border-collapse property set to collapse.
5. The header “Histogram” will precede the histogram.
6. When the user selects the **ClearHistogram** button the histogram and heading will be removed.
7. The name of the script is **histogram.php**.
8. Your script must be a self-referencing script; you may not add any other script files.
9. **Use the get method to submit your forms.**
10. If the file cannot be opened, the script will end with the message “Could not open file”.
11. The “support.php” file has the **generatePage** function that takes the body of an HTML document and generates a complete document. This is the same function presented in class. Use it to generate the final document that will be displayed. For example, if \$body has the HTML body, you will call the function as follows: ***echo generatePage(\$body);***
12. An example of executing the script is provided below. The **data.txt** file has the following data:

1
2
7
3
10
4

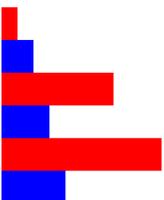
Before Selecting Any Buttons

Filename:

After Pressing the *PrintHistogram* Button

Filename:

Histogram



PAGE FOR YOUR CODE

PAGE FOR YOUR CODE

PAGE FOR YOUR CODE

Problem #4 (PHP Coding)

For this problem you will define **two** PHP scripts that support processing a payment. For this problem:

1. Define a script called **processPayment.php** that will display the following elements:
 - a. A text field that will allow users to specify an account.
 - b. A text field that will allow users to enter a payment (numeric value).
 - c. A submit button.
2. The **processPayment.php** script will verify the account number has a valid format and that the payment is a number. A valid account has the string "A-" followed by any set of characters. You can assume that if "A-" is present, a value will followed the "-". You can verify whether a payment is valid by using the PHP function **is_numeric**. First, check whether an account is valid; if the account is invalid the message "Invalid account" will be printed and no further processing will take place. If the account is valid, the payment value will be verified and if it is invalid the message "Invalid payment" will be printed; no further processing will take place. If a valid account and valid payment have been provided, the script will call the script **verify.php** passing to the script the account and payment values using sessions; **you must use sessions to pass these two values**.
3. The **verify.php** script will print the message:

**Select OK to submit payment of <PAYMENT_HERE> for account <ACCOUNT_HERE>
Select UPDATE to update payment/account**

where <PAYMENT_HERE> and <ACCOUNT_HERE> represent the payment and account, respectively.

After the above message, the buttons **OK** and **UPDATE** will be displayed. If the user selects the OK button, the message "Thank you" (by itself, nothing else will appear) will be displayed. If the user selects the UPDATE button, the **processPayment.php** script will be executed again.

4. **You MAY NOT add a third script.**
5. **Use the post method to submit your forms.**
6. The "support.php" file has the **generatePage** function that takes the body of an HTML document and generates a complete document. This is the same function presented in class. Use it to generate the final document that will be displayed. For example, if \$body has the HTML body, you will call the function as follows: *echo generatePage(\$body);*

<p><u>Before Entering Data (Step 1)</u></p> <p>Account: <input type="text"/> Payment: <input type="text"/> <input type="submit" value="Submit"/></p> <p><u>After Entering Invalid Account (Step 2)</u></p> <p>Account: <input type="text" value="B-1020"/> Payment: <input type="text" value="34.50"/> <input type="submit" value="Submit"/></p> <p>Invalid account</p>	<p><u>After Entering Valid Data (Step 3)</u></p> <p>Select OK to submit payment of 34.50 for account A-1020 Select UPDATE to update payment/account</p> <p><input type="button" value="OK"/> <input type="button" value="UPDATE"/></p> <p><u>After Selecting OK (Step 4, OK)</u></p> <p>Thank you</p> <p><u>After Selecting UPDATE (Step 4, UPDATE)</u></p> <p>Account: <input type="text"/> Payment: <input type="text"/> <input type="submit" value="Submit"/></p>
--	---

PAGE FOR YOUR CODE

PAGE FOR YOUR CODE

PAGE FOR YOUR CODE