



University of Maryland College Park

Dept of Computer Science

CMSC389N Summer 2017

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

Grader Use Only

#1	Problem #1 (Miscellaneous)	(60)	
#2	Problem #2 (PHP Coding I)	(30)	
#3	Problem #3 (PHP Coding II)	(50)	
#4	Problem #4 (PHP Coding III)	(60)	
Total	Total	(200)	

Problem #1 (HTML/CSS Concepts)

1. (3 pts) After we execute a PHP script if we execute a “View Page Source” in the browser we will see:
- Only PHP code associated with the script.
 - HTML resulting from processing the script.
 - All the contents of the file.
 - None of the above

Answer: b.

2. (3 pts) Which of the following apply to attributes of an HTML tag? Circle all that apply.
- Attributes can appear in the start tag.
 - Attributes can appear in the end tag.
 - Order in which tags appear is significant.
 - None of the above.

Answer: a.

3. (3 pts) Which of the following can replace HERE in the following link definition? Circle all that apply.

`Page`

- "http://www.cs.umd.edu"
- "www.cs.umd.edu"
- None of the above.

Answer: a.

4. (3 pts) One type of display value we saw in class is block. Which of the following apply to an element that has a block display value?
- The element begins on a new line.
 - The element takes only as much space as necessary (instead of stretching to the left and right as far as possible).
 - The element will have a red border.
 - None of the above.

Answer: a.

5. (3 pts) What is the effect associated with the following code?

`<!--#include virtual="spec.html" -->`

- The above is an HTML comment.
- Includes the contents of the spec.html file in the file where the code is found.
- To include a link to the specified file.
- None of the above.

Answer: b.

6. (3 pts) Which of the following applies to echo (PHP)? Circle all that apply.

- a. Returns a value.
- b. Print expressions separated by commas.
- c. Can print only expressions in double quotes.
- d. None of the above.

Answer: b.

7. (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.

Answer: a., c.

8. (3 pts) What is the output generated by the following PHP code?

```
if ("105" == '105') {  
    print("One");  
} else {  
    print("Two");  
}
```

Answer: One

9. (3 pts) Define a PHP constant called **AREA** that has as value 65.5.

Answer: define("AREA", 65.5);

10. (3 pts) Complete the definition of the process() function so it prints the value of the variable \$count.

```
$count = 20;  
  
function process() {  
  
}
```

Answer: global(\$count) # must appear inside of the function

11. (3 pts) Write a single PHP statement that will delete the first element of the array \$data.

Answer: unset(\$data[0]) OR \$data[0] = null

12. (3 pts) Using a single statement, initialize \$array with an array that will allow the code below to print the output:

```
Array ( [0] => John [1] => 10 [2] => 30.00 )
```

```
$str = "John, 10, 30.00";
```

```
$array =
```

```
print_r($array);
```

Answer: `explode(" ", $str);`

13. (3 pts) Initialize the string variable \$message using a heredoc. The contents of the variable will be:

```
<strong>Maryland "Terps"</strong>
```

```
$message =
```

```
If we execute echo $message we will see Maryland "Terps"
```

Answer:

```
$message = <<<DATA
```

```
<strong>Maryland "Terps"</strong>
```

```
DATA;
```

14. (3 pts) Which of the following variables are considered empty? Circle all that apply.

```
$a = 0.0;
```

```
$b = 0;
```

```
$c = "false";
```

```
$d = "";
```

Answer: \$a, \$b, \$d

15. (3 pts) Complete the following function prototype with a type hint that specifies the function returns a string (including also NULL).

```
function getInfo()
```

Answer: `function getInfo() ?string`

16. (3 pts) Write a SQL command that will create a table named “taInfo” that has the fields course (string 5 characters), taName (string 20 characters) and hours (integer).

Answer: `create table taInfo (course varchar(5), taName varchar(20), hours int);`

17. (3 pts) Write a SQL command that will insert a record into the “taInfo” table above for a ta where the course is “cs389”, the ta’s name is “Robert”, and the number of hours is 20.

Answer: `insert into taInfo values ("cs389", "Robert", 20);`

18. (3 pts) Write a SQL command that will display the name of tas that are associated with a minimum of 20 hours.

Answer: select taName from taInfo where hours >= 20;

19. (6 pts) Define a CSS class selector called **myStyle** that defines blue and 4em as the text color and font size, respectively.

Answer:

```
.myStyle {  
    color: blue;  
    font-size: 4em;  
}
```

Problem #2 (PHP Coding)

Define two PHP classes called **Beverage** and **Coffee**. The specification for these classes are:

1. **Beverage** class

- The class has two private instance variables called **\$name** and **\$calories**.
- A static variable named **\$totalBeverages** keeps track of the number of **Beverage** objects created.
- A constructor that initializes a **Beverage** object. It has a name and calories as parameters.
- A toString method that prints the name and calories associated with a **Beverage** object. See the output below for format information.
- getCalories – get method for calories field.
- getTotalBeverages – static method that returns total number of **Beverage** objects created.
- Use strict types for parameters and return values. You can assume the declaration **declare(strict_types=1)** already exists.

Answer:

```
class Beverage {  
    private $name, $calories;  
    static $totalBeverages = 0;  
  
    public function __construct(string $name, int $calories) {  
        $this->name = $name;  
        $this->calories = $calories;  
        Beverage::$totalBeverages++;  
    }  
  
    public function __toString() : string {  
        return "Name: \"\".\"$this->name.\"\", Calories: \".\"$this->calories;  
    }  
  
    public function getCalories() : int {  
        return $this->calories;  
    }  
  
    public static function getTotalBeverages() : int {  
        return Beverage::$totalBeverages;  
    }  
}
```

2. Coffee class

- The class extends the **Beverage** class and has a private instance variable called **\$caffeine**.
- A constructor that has name, calories, and caffeine as parameters.
- A toString method that prints the name, calories, and caffeine associated with the **Coffee** object. See the output below for format information.
- Use strict types for parameters and return values. You can assume the declaration **declare(strict_types=1)** already exists.

Answer:

```
class Coffee extends Beverage {
    private $caffeine;

    public function __construct(string $name, int $calories, float $caffeine) {
        parent::__construct($name, $calories);
        $this->caffeine = $caffeine;
    }

    public function __toString() : string {
        $BeverageStr = parent::__toString();
        return $BeverageStr.", Caffeine: ".$this->caffeine;
    }
}
```

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

main function

```
function main() {
    $beverage1 = new Beverage("Juice", 200);
    echo $beverage1."<br>";
    $coffee1 = new Coffee("Latte", 300, 34.75);
    echo $coffee1."<br>";
}
```

Output

```
Name: "Juice", Calories: 200
Name: "Latte", Calories: 300, Caffeine: 34.75
```

Problem #3 (PHP Coding)

Write a PHP script that generates a form that prints a list with entries from a file. The specifications for this problem are:

1. Define a textfield (with default value data.txt) that allow us to enter a file name.
2. Define a textfield (with default value 2) that allow us to enter a number.
3. Two submit buttons named “PrintList” and “ClearList” will allow us to generate an ordered list or to clear/remove the list.
4. When the user selects the **PrintList** button the file specified in the file text field will be opened. The script will then read a number of entries from the file that corresponds to the number textfield value. An ordered list based on these entries will be displayed after the buttons. The heading “List” will precede the list.
5. When the user selects the **ClearList** button the list and heading (if any) will be removed.
6. The name of the script is **printList.php**.
7. Your script must be a self-referencing script; you may not add any other script files.
8. Use the post method to submit your form.
9. If the file cannot be opened the script will end with the message “Could not open file”.
10. 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);***
11. An example of executing the script is provided below. The data.txt file has the following three lines:

Windows
PC
Linux

Before Selecting Any Buttons

Filename:

Number of Entries:

After Pressing the *PrintList* Button

Filename:

Number of Entries:

List

1. Windows
2. PC

After Pressing the *ClearList* Button

Filename:

Number of Entries:

Answer:

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

$answer = "";
if (isset($_POST["printList"])) {
    $fp = fopen($_POST["filename"], "r") or die("Could not open file");
    $currentEntries = 0;
    $answer = "<h2>List</h2><ol>";
    while (!feof($fp) && $currentEntries < $_POST["entries"]) {
        $line = trim(fgets($fp));
        $answer .= "<li>$line</li>";
        $currentEntries++;
    }
    $answer .= "</ol>";
    fclose($fp);
} if (isset($_POST["clearList"])) {
    $answer = "";
}

$body = <<<EOBODY
    <form action="printList.php" method="post">
        Filename: <input type="text" name="filename" value="data.txt"><br><br>
        Number of Entries: <input type="text" name="entries" value="2"><br><br>
        <input type="submit" name="printList" value="PrintList">
        <input type="submit" name="clearList" value="ClearList">
    </form>
    <br>
    $answer
EOBODY;

echo generatePage($body);
```


Problem #4 (PHP Coding)

For this problem you will define two PHP scripts that support the generation of a table with links to resources. For this problem:

1. Define a script called **processRequest.php** that will display the following elements:
 - a. A textfield that will allow users to specify a basename for a file.
 - b. A textfield that will allow users to enter a number (limit).
 - c. A submit button.
2. The **processRequest.php** script will verify that the basename and limit are valid values. A valid basename is either the string “terps” or “umcp”. A valid limit is a value between 1 (inclusive) and 30 (inclusive). You can assume users will enter a number for the limit value. If the basename is invalid the message “Invalid basename” will be displayed below the textfields and the button; if the limit is invalid the message “Invalid limit” will be displayed. Keep in mind that if both values are invalid both invalid messages will be displayed. No further processing will take place if invalid values were provided. If valid values were provided, the script will call a second script called **genTable.php**, passing to the script the basename and limit values. **You MUST use sessions in order to pass these two values to the genTable.php script.**
3. The **genTable.php** script will generate a table with links (e.g., terps1.html). The number of links corresponds to the limit value and each link is created by concatenating the basename, a number, and the .html extension.
4. The form generated by **processRequest.php** must have “terps” and 5 as the default basename and limit, respectively. If the user enters invalid values, the invalid values must appear in the form after the submit button has been selected. A form with default values or empty textfields must not be displayed after invalid values have been provided.
5. 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);***

Before Entering Data or Selecting the Submit Button

Basename: Limit:

After Entering Invalid Data

Basename: Limit:
Invalid basename
Invalid limit

After Pressing Submit for Basename terps and Limit 5

Links

terps1.html
terps2.html
terps3.html
terps4.html
terps5.html

processRequest.php

Answer:

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

$bottom = "";
$valid = true;
if (isset($_GET["submitButton"])) {
    $basename = $_GET["basename"];
    $limit = $_GET["limit"];

    if ($basename != "terps" && $basename != "umcp") {
        $bottom .= "Invalid basename<br>";
        $valid = false;
    }
    if ($limit > 30 || $limit < 1) {
        $bottom .= "Invalid limit<br>";
        $valid = false;
    }
    if ($valid) {
        session_start();
        $_SESSION['basename'] = $basename;
        $_SESSION['limit'] = $limit;

        header("Location: genTable.php");
    }
} else {
    $basename = "terps";
    $limit = 5;
}
$body = <<<EOBODY
    <form action="$ _SERVER[PHP_SELF]" method="get">
        <strong>Basename: </strong><input type="text" name="basename"
value="$basename" />
        <strong>Limit: </strong><input type="text" name="limit" value="$limit" />
        <input type="submit" name="submitButton" />
    </form>
EOBODY;

$page = generatePage($body.$bottom);
echo $page;
```

genTable.php

Answer:

```
session_start();
$basename = trim($_SESSION['basename']);
$limit = $_SESSION['limit'];
$body = "<h2>Links</h2>";
$body .= "<table border='1'>";
for ($i = 1; $i <= $limit; $i++) {
    $body .= "<tr><td>";
    $filename = $basename.$i.".html";
    $body .= "<a href=\"\$filename\">$filename</a>";
    $body .= "</td></tr>";
}
$body .= "</table>";
$page = generatePage($body);
echo $page;
```