COWB40385 Introduction to Web Programming

# Introduction to Web Programming Assessment Part 2

## Preliminary

Each exercise below should be completed as a separate web page. Once you have completed Exercise One you should make a copy of the web page you have produced and use that copy as the basis for Exercise Two. This process should be repeated until completion of the final exercise.

All web pages should use, and be compliant with, the following doctype:

<!DOCTYPE html>

All functionality should take place on the server and be written in PHP.

## Exercise One – Multiplication Table

Using two loops, output a table showing the multiplication tables up to 10x10 (as default). Your table should appear similar to that shown in Figure 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

Figure 1: Example multiplication table layout

Your code should allow the user to specify the maximum number for which a table row/column is calculated (you will need to use a form).

## Exercise Two – Basic Output Grade Point Scale Converter

Create a PHP form that will accept a percentage grade and convert it to the appropriate grade point. There are separate scales for undergraduate awards (BSc, HND/C) and postgraduate awards (MSc, MRes). Your form should accept a percentage value between 1-100% and what level the grade belongs to. An example form is shown in Figure 2. The form should output the grade point (1-15) and the Classification.

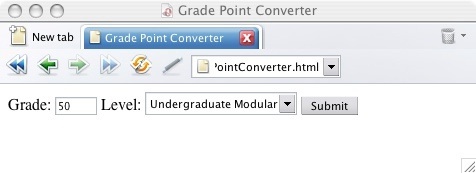


Figure 2 Example grade point converter form

### Undergraduate Modular Framework Awards

70%+ Grade Points 13 - 15 First Class Honours

60 - 69% Grade Points 10 - 12 Upper Second Class Honours

50 - 59% Grade Points 7 - 9 Lower Second Class Honours

40 - 49% Grade Points 4 - 6 Third Class Honours

0 - 39% Grade Points 0 - 3 Fail

### HND - Current Scheme

70%+ Distinction - grade points 13 - 15

53 - 69% Merit - grade points 8 - 12

40 - 52% Pass - grade points 4 - 7

0 - 39% Fail - grade points 1 - 3

### Masters

70%+ Distinction - grade points 13 - 15

60 - 69% Pass with Merit - grade points 10 - 12

50 - 59% Pass - grade points 7 - 9

0 - 49% Failure - grade points 1 – 6

## Exercise Three – Number Guessing Game

Create a form using PHP that accepts guesses of a number between 1 and 100 inclusive. Each guess should be matched against a randomised number (between 1 and 100 inclusive) which is set for the duration of the game. The game should output whether the number guessed is less than, equal to (you’ve won) or more than the number to be guessed. Once the number has been correctly guessed the game ends with the option of starting another game.

## Exercise 4 – Variable Input

Create a form that can accept three string values as a default but allows the user to choose as many or as few as they like. If the user chooses a different number of inputs then a new form should be generated accordingly when the user submits their choice.

Upon posting this form with completed values, have the contents of the fields placed into an array. Output the array to the browser, sort ascending and output then sort descending and output. You may sort the array in any way you see fit.

Note: For convenience, PHP allows you to name your input fields with [] appended to their name. Posting these fields will place all the values into an array. For example:

<input type="text" name="names[]" id="txt1" />

<input type="text" name="names[]" id="txt2" />

If the form uses the *POST* method, the values that the user enters can be accessed with the following PHP code:

$names = $\_POST['names'];

This code produces an array called *$names* with two values stored in it corresponding to the text boxes submitted by the form.

# Assessment Criteria

## Contribution of Parts 1, 2 and 3 to the Coursework

Assessment Part 1, Assessment Part 2 and Assessment Part 3 will be handed out between weeks 2 and 4 inclusive. The three parts constitute 40% of the overall mark for the coursework. The coursework constitutes 100% of the marks available for the module.

Of the 40% contributed by Parts 1, 2 and 3 the individual allocation is as follows:

Part 1: 10%

Part 2: 14% (this part)

Part 3: 16%

**Total: 40%**

## Marks Allocation for Assessment Part 2

### Functionality marks (worth 80% of the marks available for Part 1)

2=Functionality completed, 1=Good attempt, 0=Poor attempt/No attempt

|  |  |  |
| --- | --- | --- |
|  | **Functionality** | **Marks (out of)** |
| Exercise 1 | Two loops used effectively | 2 |
| User specifies first number (**a** multiplied by b) | 2 |
| User specifies second number (a multiplied by **b**) | 2 |
| Output of the multiplication tables | 2 |
| Output using the HTML table elements | 2 |
| Exercise 2 | User specifies grade (percentage) | 2 |
| Value between 1 and 100 inclusive | 2 |
| Output grade point and classification message | 2 |
| Functionality provided for 2 award structures | 2 |
| Functionality provided for 3 award structures | 2 |
| Exercise 3 | User enters guess | 2 |
| User guess between 1 and 100 inclusive | 2 |
| Randomise function used | 2 |
| Random number between 1 and 100 inclusive | 2 |
| Random number persisted for game duration (visible for demonstration purposes) | 2 |
| Identify guess greater than random number | 2 |
| Identify guess less than random number | 2 |
| Identify guess equal to random number | 2 |
| Game completes (option to restart) | 2 |
| Exercise 4 | Form shown with three text boxes as default | 2 |
| User specifies number of text boxes | 2 |
| Form re-generated for number of text boxes specified by user | 2 |
| Form contents placed in array | 2 |
| Array sorted alphabetically | 2 |
| Output sorted alphabetical list | 2 |

### Discretionary Mark (worth an additional 20% of the marks available for Part 2)

There will be a further mark available as the discretion of the assessor to be awarded for the adoption of best practices as evidenced in the PHP code.

|  |  |
| --- | --- |
| **Function/Requirement** | **Mark (out of)** |
| Adoption of ‘best practice’ PHP development | 100 |

70+

Exceptional code of a professional standard that is clearly written (indented, variables and functions well named, follows a naming convention) using a consistent style with little or no duplication and is well documented (code commenting).

60-69

Very good code that is clearly written (indented, variables and functions well named, follows a naming convention) using a consistent style with little or no duplication and has some documentation (code commenting).

50-59

Very good code that is clearly written (indented with good naming convention) some documentation (code commenting).