Web Scripting for Applications Revision Pack

The exam is 3 hours and contains 2 parts. This is the first year that Web Scripting for Applications has run as a distinct subject, so there are no previous papers to look at. Questions shown here have been taken from the previous second year programming module and are a good indication of the style and type of questions that will occur.

Exam structure

Part A contains 4 questions which must *all* be completed – two questions worth 10 marks and two questions worth 15 marks. These are questions that typically require writing, debugging or explaining PHP code.

Part B contains 4 essay style questions worth 25 marks each, from which you must choose 2.

Part A style questions

One

The HTML markup for a form is shown below. It is intended that a student will enter feedback about the course they are studying (indicating their name, course title, modules being studied, course rating and other comments).

Write a PHP response page that will print this information to the screen as HTML. This should show the student name, the full course title, a list of the modules that the student is studying, the course rating and any other comments.

```
<form name="feedback" action="response.php" method="POST">
<p>Enter your name: <input type="text" name="studentName"></p>
<p>Course Title
    <select name="courseTitle">
      <option value="BIT">BSc Business Information Technology</option>
      <option value="AIS">BSc Accounting and IS</option>
      <option value="BIM">BSc Business Information Management</option>
    </select>
</p>
<p>Select the modules you are taking:<br />
Business Statistics <input type="checkbox" name="modules[]" value="Business Stats"
/><br />
Web Scripting <input type="checkbox" name="modules[]" value="Web scripting" /><br
/>
Marketing  <input type="checkbox" name="modules[]" value="Marketing" /><br />
</p>
<p>How do you rate the course:
    <input type="radio" name="rating" value="Outstanding" />Outstanding
    <input type="radio" name="rating" value="MostlyOK" />Mostly Adequate
    <input type="radio" name="rating" value="Dull" />Dull
</p>
<p>Other Comments: <textarea name="comments" cols="60" rows="3"></textarea></p>
<p>Thank you for your feedback <input type="submit" name="continue"
value="Continue" /></p>
</form>
```

Two

A single table called student, inside a mysql database called feedback, hosted on localhost, is used to collate student feedback results. Using the SQL statement “select studentname, coursetitle, rating, comments from student”, write a PHP page which will extract the records as an associative array and then print them to screen.

The following built-in functions may be useful:

```
mysql_connect(name) - Connects to a mysql server hosted on name
mysql_select_db(databasename) – Opens a connection to databasename
mysql_query(SQL statement) – returns the results of running the SQL statement on the
previously opened database as the name of an associative array
mysql_fetch_array(name) – Iterates through a named associative array, returning one row
at a time until no more rows are left.
```

Three

A PHP function countResponses is to be defined that will take an array of feedback statements and return the number of times that a particular statement has occurred.

```
$responses=array('Outstanding','Outstanding','MostlyOK','Dull','Outstanding');
print "No is ".countResponse('Outstanding',$responses);
```

Complete the definition of countResponse below.

```
function countResponse(                                    ){
$countNumber=0;
foreach(                                 ){
  if(                          ) {                                  };
  };
return                           ;
}
```

Four

The HTML markup for a form is shown below. It is intended that a user will enter feedback about a film they have seen (indicating their name, film title, location of cinema, film rating and other comments).

Write a PHP response page that will print this information to the screen as HTML. This should show the users name, the full film title, the cinema where the film was viewed, the film rating and any other comments. If the cancel button is used, the response page should state that no entry was submitted and terminate.

```
<form name="feedback" action="response.php" method="POST">
<p>Enter your name: <input type="text" name="name"></p>
<p>Film Title
    <select name="filmTitle">
      <option value="GC">The Golden Compass</option>
      <option value="IAL">I Am Legend</option>
      <option value="CWW">Charlie Wilsons War</option>
    </select>
</p>
<p>Select the cinema where you saw the film<br />
Richmond <input type="checkbox" name="cinema[]" value="Richmond" /><br />
Kingston <input type="checkbox" name="cinema[]" value="Kingston" /><br />
Feltham  <input type="checkbox" name="cinema[]" value="Feltham" /><br />
</p>
<p>How do you rate the film:
    <input type="radio" name="rating" value="Outstanding" />Outstanding
    <input type="radio" name="rating" value="Average" />Average
    <input type="radio" name="rating" value="Dull" />Dull
</p>

<p>Other Comments: <textarea name="comments" cols="60" rows="3"></textarea></p>
<p>Thank you for your feedback <input type="submit" name="continue" value="Cancel"
/>
<input type="submit" name="continue" value="submit" />
</p>
</form>
```

Five

A single table called filmrating, inside a mysql database called film, hosted on localhost, is used to collate film feedback results. Using the SQL statement "select name, filmtitle, rating, comments from filmrating", write a PHP page which will extract the records as an associative array and then print them to screen. Ensure that the code terminates if any of the database connectivity functions fail.

The following built-in functions may be useful:

```
resourcename = mysql_connect(name) - Returns a connection to a mysql server hosted
on name as a PHP resource
name = mysql_select_db(databasename) - Opens a connection to databasename,
returning a boolean
resultresource = mysql_query(SQL statement) - returns the results of running the
SQL statement on the previously opened database as a PHP resource
arrayname = mysql_fetch_array(name) - Iterates through a named associative array,
returning a one row array at a time until no more rows are left
```

Six

A PHP function countResponses has been defined that will take an array of film titles with feedback comments and return the number of times that a particular feedback comment has occurred. Use appropriate diagrams to explain how the function works and the values returned.

countRatings is defined as follows:

```
function countRatings($searchTerm,$responseArray){
foreach($responseArray as $item){
  if ($item[1]==$searchTerm) $countArray[$item[0]]++;
  };
return $countArray;
}
```

The function is used as follows:

```
$response1=array("I Am legend","Outstanding");
$response2=array("I Am legend","Average");
$response3=array("Charlie Wilsons War","Outstanding");
$response4=array("Charlie Wilsons War","Outstanding");
$response5=array("The Golden Compass","Dull");
$responses=array($response1,$response2,$response3,$response4,$response5);

$result=countRatings('Outstanding',$responses);

foreach($result as $filmtitle=>$item){
  print $filmtitle." scored ".$item." Outstanding<br />";
  };
```

Seven

The HTML markup for a form is shown below. It is intended that a user will request a film they wish to book (indicating the film name, type of seat, their own name, and an indication of whether they have visited the cinema before).

Write a PHP response page that will print this information to the screen as HTML. This should show the users name, the full film title, the seat type selected, and whether they have visited the cinema before. If the cancel button is used, the response page should state that no entry was submitted and terminate.

```
<form name="form1" action="response.php" method="POST">
<p>Film Title:
    <select name="filmName">
      <option value="HP">Harry Potter</option>
      <option value="ST">Star Trek</option>
      <option value="Watch">Watchmen</option>
    </select>
</p>
<p>Which seat type would you like:
    <input type="radio" name="seatType" value="premier" />Premier
    <input type="radio" name="seatType" value="circle" />Circle
    <input type="radio" name="seatType" value="stall" />Stall
</p>

<p>Your name:<input type="text" name="clientName"></p>
<p>I have visited this cinema before <input type="checkbox" name="returningClient"
value="checkbox" /></p>
<p>Thank you for your feedback <input type="submit" name="continue"
value="Continue" />
<input type="submit" name="continue" value="Cancel" /></p>
</form>
```

Eight

A single table called seatAllocation, inside a mysql database called booking, hosted on localhost, is used to collate cinema seat bookings. Using the SQL statement "select filmname, date, time, count(seatType) from seatAllocation groupby seatType", write a PHP page which will extract the records as an associative array and then print them to screen. Ensure that the code terminates if any of the database connectivity functions fail.

The following built-in functions may be useful:

```
resourcename = mysql_connect(name) - Returns a connection to a mysql server hosted
on name as a PHP resource
name = mysql_select_db(databasename) - Opens a connection to databasename,
returning a boolean
resultresource = mysql_query(SQL statement) - returns the results of running the
SQL statement on the previously opened database as a PHP resource
arrayname = mysql_fetch_array(name) - Iterates through a named associative array,
returning a one row array at a time until no more rows are left
```

Nine

A PHP function countSeats has been defined that will take an array of film titles, seat types and numbers of seats sold and return the total number of seats sold. Use appropriate diagrams to explain the structure of the associative array, how the function works and the value returned.

countSeats is defined as follows:

```
function countSeats($filmName, $seatType, $responseArray){
$seatsSold=0;
foreach($responseArray as $item){
    if ($item["filmName"]==$filmName and $item["seatType"]==$seatType)
$seatsSold=$seatsSold+$item["noOfSeats"];
    };
return $seatsSold;
}
```

The function is used as follows:

```
$seatsBooked1=array( "filmName"=>"Watchmen", "seatType"=>"Premier", "noOfSeats"=>5,
"session"=>"Morning");
```

```
$seatsBooked2=array( "filmName"=>"Watchmen", "seatType"=>"Circle", "noOfSeats"=>12,
"session"=>"Morning");
$seatsBooked3=array( "filmName"=>"Watchmen", "seatType"=>"Standard", "noOfSeats"=>5,
"session"=>"Morning");
$seatsBooked4=array( "filmName"=>"Watchmen", "seatType"=>"Premier", "noOfSeats"=>7,
"session"=>"Afternoon");
$seatsBooked5=array( "filmName"=>"Watchmen", "seatType"=>"Circle", "noOfSeats"=>10,
"session"=>"Afternoon");
$seatsBooked6=array( "filmName"=>"Watchmen", "seatType"=>"Standard", "noOfSeats"=>9,
"session"=>"Afternoon");

$seatsBooked=array($seatsBooked1,$seatsBooked2,$seatsBooked3,$seatsBooked4,$seatsBo
oked5,$seatsBooked6);

$result=countSeats("Watchmen","Premier", $seatsBooked);

print "Number of seats sold was $result";
```

Ten

The HTML markup for a self-validating form is shown below. Users can use the form to register on a site by supplying a desired username, an email address and confirming the terms and conditions of the site.

Explain how the form works.

```
<?php
if ( !$_POST['username'] or !$_POST['email'] or !$_POST['terms'] ) {
print('<html>');
print('<head>');
print('<title>Site Registration</title>');
print('</head>');
print('<body>');
print('<h1>Site Registration</h1>');
print('<p>This form will allow you to register on this site</p>');
print('<h2>Please enter your details</h2>');
print('<form name="form1" method="post" action="form.php">');

print('<p>Enter your desired username: <input type="text" name="username"
value="'.$_POST['username'].'"> ');
if (!$_POST['username']) {print ('<i>Please enter a username</i>'); };
print('</p>');

print('<p>Enter your email address: <input type="text" name="email"
value="'.$_POST['email'].'"> ');
if (!$_POST['email']) {print ('<i>Please enter a valid email address</i>'); };
print('</p>');

print('<p>I acknowledge the terms and conditions of the site: <input
type="checkbox" name="terms" value="checked" /> ');
if (!$_POST['terms']) {print ('<i>Please select to confirm site terms and
conditions</i>'); };
print('</p>');

print('<p>Press to complete registration <input type="submit" name="continue"
value="Continue" /></p>');

print('</form>');
print('</body>');
print('</html>');
}
else
{
    print('<html>');
    print('<body>');
    print('Form processed');
    /* Rest of form processing routine */

    print('</body>');
    print('</html>');
};
?>
```

Eleven

A database table called user, inside a mysql database called siteReg, hosted on localhost, is used to store user registration details, which must be unique. Use appropriate SQL statements to write a PHP page which will look to see if a supplied username ($_POST['username']) or a supplied email address ($_POST['email']) is already present in the table and then write the results of the search to the screen.

Ensure that the code terminates if any of the database connectivity functions fail.

The user table was created with the following SQL:

```
CREATE TABLE `user` (
`userID` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`username` VARCHAR( 20 ) NOT NULL ,
`email` VARCHAR( 30 ) NOT NULL ,
`password` VARCHAR( 20 ) NOT NULL
);
```

The following built-in functions may be useful:

```
resourcename = mysql_connect(name) - Returns a connection to a mysql server hosted
on name as a PHP resource or false if no connection possible
name = mysql_select_db(databasename) - Returns an connection to databasename, or
false if no connection possible
resultresource = mysql_query(SQL statement) - returns the results of running the
SQL statement on the previously opened database as a PHP resource
arrayname = mysql_fetch_array(name) - Returns the next unprocessed row from the
named PHP resource as an associative array, or false if there are no rows left
```

Twelve

A PHP function countActivities has been defined, that will take an array of usernames, activity types and time of the day (Morning, Afternoon and Evening) when a user is logged in. The function will search this array for matching usernames and activities and then return a count of these occurrences as an associative array. Identify the errors in the function (there are at least 10) and rewrite the code so that it works.

countActivities is defined as follows:

```
function countActivities($username, $activity, $response){
for($responseArray as $item){
if ($Item["username"]==$username and $Item["activity"]==$response) {
    choice ($item["session"]) {
        case 'Morning':
            $totals[Morning]+1;
            break;
        case 'Afternoon':
            $totals["Afternoon"]+1;
        case 'Evening':
            $totals["Evening"]++;
            break;
        };
};
};

The function is used as follows:

$activity1=array( "username"=>"fred", "activity"=>"changePassword",
"session"=>"Morning");
$activity2=array( "username"=>"fred", "activity"=>"browse", "session"=>"Morning");
```

```
$activity3=array( "username"=>"fred", "activity"=>"browse",
"session"=>"Afternoon");
$activity4=array( "username"=>"fred", "activity"=>"postComment",
"session"=>"Afternoon");
$activity5=array( "username"=>"fred", "activity"=>"postComment",
"session"=>"Evening");
$activity6=array( "username"=>"fred", "activity"=>"postComment",
"session"=>"Evening");

$activities=array($activity1,$activity2,$activity3,$activity4,$activity5,$activity6
);

$result=countActivities("fred","postComment", $activities);

print "Number of browse activities for Fred was";
print "Morning count was ".$result['Morning'];
print "Afternoon count was ".$result['Afternoon'];
print "Evening count was ".$result['Evening'];
```

Part B style questions

One

A PHP session can be used to authenticate a user and allow privileged access to particular web pages. Use extracts of PHP code to explain an example session with interaction between the browser and server to perform login, login authentication, failed authentication and logout.

Two

The three-tier development architecture has been the traditional layering for web application development. Use PHP examples to explain the three layers, how they interact and how the MVC design pattern can be used to support this architecture.

Three

PHP sessions can be used to store state information across web pages. Use appropriate examples to demonstrate how PHP sessions work, along with the kinds of web applications that can use this technique.

Four

PHP applications can be written using layers of presentation logic, application logic and database logic. Use PHP examples to explain the content of the three layers, how they interact and how classes and objects can be used to support this development architecture.

Five

PHP sessions can be used to create secure areas by storing authentication information across web pages. Use appropriate examples to demonstrate how this works.

Six

The MVC design pattern can support the traditional layering of web applications into presentation logic, application logic and database logic. Use PHP examples to explain how this works, which content falls where and how classes and objects can be used to support this development architecture.

Seven

A web form is to be designed to capture user registration details for a site. If *username*, *email address* or *acknowledgement* of the terms and conditions are missing, the form is displayed again, incorporating a message that indicates the missing fields. Write PHP code and XHTML to implement the form.

Eight

PHP sessions can be used to create secure areas by storing authentication information across web pages. Use appropriate examples to demonstrate how this works.

Nine

PHP can be used to make a rudimentary templating system that assists in HTML deployment and reuse across multiple web site pages. Use examples to explain how such a system can be implemented. Show how a more advanced system would use classes and objects.