OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

SAMPLE EXAM PAPER 2013

CS1109 Programming and Web Development

Dr. D.G. Bridge

Answer **all** questions.
Silent non-programmable calculators may be used.

Time allowed: Three hours

1. (*25%*)

i) (*5%*) Evaluate each of the following as PHP would, and state the *value* and *type* of `$x` after each statement:

a. `$x = 2 - 3 + 4;`

b. `$x = 8 + 3 % 3;`

c. `$x = 'abc'. (2 + 3) . 'def';`

d. `$x = 2 > 3 || (7 < 8 && 5 != 6);`

e. `$x = (2 > 3 ? 10 : (5 <= 6 ? 20 : 30));`

ii) (*5%*) Suppose that variable `$x` contains an integer, and consider the following:

```
if ( $x > 3 )
{
    echo "fizz";
}
elseif ( $x > 7 )
{
    echo "buzz";
}
if ( $x > 11 )
{
    if ( $x < 15 )
    {
        echo "buzz";
    }
}
```

Precisely state for what range of values the output will be

a. just "fizz"?

b. just "buzz"?

c. both "fizz" and "buzz"?

d. neither?

iii) (*5%*) Consider the following function, whose parameter `$a` is an array:

```
function foo( &$a )
{
    $x = 0;
    foreach ( $a as $v )
    {
        $x++;
    }
    return $x;
}
```

a. Describe what it computes and returns.

b. Suggest a more appropriate name for the function.

iv) (*5%*) Consider the following function, whose parameter `$n` is an integer, and which is intended to compute the sum of all positive *odd* numbers up to and including `$n` (or up to and including one less than `$n` if `$n` is even):

```
function sum_odds( $n )
{
    $sum = 0;
    if ( ($n % 2) == 0 ) // If $n is even, make it odd
    {
        $n--;
    }
    while ( $n > 0 )
    {
        $sum += $n;
        $n--;
    }
    return $sum;
}
```

The function has no compile-time or run-time errors, but it does have a logic error.

a. Explain the error. Use an example, if you wish.

b. Explain how to fix the error.

v) (*5%*) Compare this snippet of PHP:

```
$i = (int) $_GET['number'];
while ( $i > 0 )
{
    $i--;
    echo $i;
}
```

with this one:

```
for ( $i = (int) $_GET['number']; $i > 0; $i— )
{
    echo $i;
}
```

If you think they are equivalent (i.e. same outputs for the same inputs), state which one is more efficient and why it is the more efficient. If you think they are not equivalent, how will their outputs differ?

2. (*25%*) Households that are thinking about installing solar panels want to estimate how long it will take for the benefit of the panels to equal the installation cost. They enter into the following form the installation cost, the size of their panels (small, medium, or large), and the amount of power (in kiloWatts) they typically consume each year:

```
<form action="breakeven.php" method="get">

    <input type="text" name="install_cost_euros" placeholder="0" />

    <input type="radio" name="size" value="small" />
    <input type="radio" name="size" value="medium" />
    <input type="radio" name="size" value="large" />

    <input type="text" name="annual_consumption_kW" placeholder="0" />

    <input type="submit" />

</form>
```

Write `breakeven.php` which takes data from this form and uses the following information to determine how many years the panels must operate before their total benefit (i.e. savings plus receipts) pays off their installation cost:

- In the first year of operation, small panels generate 1500kW, medium panels generate 2000kW, and large panels generate 2200kW.
- In each subsequent year of operation, panels generate 95% of what they generated the previous year. For example, in the second year a small panel generates 1425kW (95% of 1500); in their third year of operation, they generate 1353.75kW (95% of 1425); and so on.
- A household *saves* on its fuel bill 0.15 euros for each kW that it generates and consumes.
- A household *receives* 0.05 euros for each kW that it exports to the national grid, i.e. each kW that it generates but does not consume.

A stylesheet is not required. Validation of user data is not required. You can ignore the possibility that the program is infinite — where the benefits never manage to cover the costs.

Marks will be awarded for correctness, elegance and efficiency.

3. (*25%*) In your answers to this question, you may use the following built-in functions but no others:
   - The `array` function for creating an array, e.g. `array()` creates an empty array.
   - The `in_array` function, whereby **`in_array`**`( $v, $vs )` returns **`true`** if `$v` is an element in array `$vs`; otherwise, it returns **`false`**. For example, it returns **`true`** if `$v` is 7 and `$vs` is `array(3, 1, 7, 13)`.

   A group of friends use an associative array to record their names and the number of hands of poker each has won, e.g.:

```
array("Baz" => 0, "Liz" => 3, "Jen" => 2, "Dan" => 2, "Guy" => 0)
```

   i) (*4%*) Define a PHP function called `get_num_hands` which takes in an array like the one above and returns how many hands have been played, e.g. 7 in the example above.

   ii) (*8%*) Define a PHP function called `get_wins` which takes in an array like the one above and returns an indexed array of its values but with no duplicates, e.g. `array(0, 3, 2)` in the example above.

   iii) (*13%*) Define a PHP function called `get_league` which takes in an array like the one above and returns a 2-dimensional array like the following:

```
array
(
    array("Baz", "Guy"),
    array(),
    array("Jen", "Dan"),
    array("Liz"),
    array(),
    array(),
    array(),
    array()
)
```

   This is an indexed array. In position 0 is an array of people who have won no games; in position 1 is an array of people who have won one game; and so on up to the total number of games played. (You may use the `get_num_hands` functions you defined in part (3i) above, if you wish.)

4. (*25%*) EmuAirways uses the following two database tables to record the flights it offers and the airports it uses:

flights		
flight_num	going_from	going_to
ea101	ORK	STN
ea102	ORK	EDI
ea103	ORK	GLA
ea104	ORK	CDG
ea105	STN	CDG
ea106	STN	BER
...	...	...

airports	
airport	country
ORK	Ireland
STN	England
EDI	Scotland
GLA	Scotland
CDG	France
BER	Germany
...	...

Suppose `destinations.html` contains this form:

```
<form action="destinations.php" method="get">
    <input type="text" name="going_from" />
    <input type="submit" />
</form>
```

The user enters the code of the airport from which s/he likes to travel. Write `destinations.php` which receives the user's input, queries the database, and outputs a well-formed HTML unordered list of countries to which there is at least one direct flight from the user's chosen airport, including nested lists of flight numbers to that country.

For example, using the database shown above, if the user enters "ORK", the user's browser will display something like the following:

- England
  - ea101
- France
  - ea104
- Scotland
  - ea102
  - ea103

You may assume that the following function invocation will connect your script to the database:

```
$dbconnection = mysqli_connect( 'my_server', 'me', 'my_passsword', 'my_db' );
```

You may use any built-in functions that you wish, including: `mysqli_query( $dbconnection, $sql )` to execute query `$sql` on database connection `$dbconnection`; `mysqli_num_rows( $dbresult )` to find out how many rows the result-set `$dbresult` contains; and `mysqli_fetch_assoc( $dbresult )`, which returns the next row of the result-set or `NULL` when there are no more rows.

A stylesheet is not required. Validation of user data is not required.

Marks will be awarded for correctness, elegance and efficiency.