OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

2015/2016
SEMESTER 2 — SUMMER 2016

CS1116 Web Development II

Dr. Helen Purchase,
Professor C. Sreenan,
Dr. D.G. Bridge

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

Time allowed: 90 minutes
(Suggestion: Q1 54 minutes; Q2 36 minutes)

PLEASE DO NOT TURN THIS PAGE UNTIL INSTRUCTED TO DO SO.
PLEASE ENSURE THAT YOU HAVE THE CORRECT EXAM PAPER

1. (*45 marks*) Betty is a committed marathon runner who uses the following database table (shown with sample data) to record the dates of the marathons she has run and how long she took to complete them (in minutes):

marathons	
marathon_date	run_time
2013-06-04	230
2014-04-21	224
2014-11-03	237
2015-06-02	224
2015-10-27	224

Betty desires a Python server-side program called `insert.py` that implements a *self-processing page* as follows:

- If the user submits no data, the program outputs the form.
- Otherwise, the program inserts the new date and time into the database and then outputs the form followed by a message that says that the insertion was successful.

In *both* cases, the program also outputs an HTML table of all of Betty's marathons in order of date. Each row of the HTML table contains the date of the marathon, how long it took Betty to run the marathon, and either 'Y' if this is a marathon in which Betty ran her best (lowest) time or 'N' otherwise. Here is an example of the HTML table for the sample database contents shown earlier:

Date	**Run time**	**Best**
2013-06-04	230	N
2014-04-21	224	Y
2014-11-03	237	N
2015-06-02	224	Y
2015-10-27	224	Y

However, if the program encounters problems communicating with the database management system, the program outputs the form followed by a message of apology.

Complete `insert.py`, which is shown on the next page, by giving Python statements to replace the comment in the middle of the program.

Notes:

- You may assume that the following will create a connection to the database management software:

  ```
  connection = db.connect('my_server', 'me', 'my_password', 'my_db')
  ```
- There is credit for guarding against JavaScript injection attacks and SQL injection attacks.
- There is *no* credit for validating the user's input and *no* credit for making the form sticky. You are therefore advised to omit these from your solution.

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from cgi import FieldStorage, escape
import pymysql as db

# Your code would appear here

print("""
    <!DOCTYPE html>
        <html lang="en">
            <head>
                <title>Insert a marathon</title>
            </head>
            <body>
                <form action="insert.py" method="post">
                    <label>Marathon date (YYYY-MM-DD): </label>
                    <input type="text" name="marathon_date"><br>
                    <label>Run time: </label>
                    <input type="text" name="run_time"><br>
                    <input type="submit" value="Insert">
                </form>
                %s
            </body>
        </html>""" % (result))
```

2. (*30 marks*) A web page uses a JavaScript program to run a simple 'game' in a 500 × 300 pixel canvas.

   The JavaScript program creates ten 'critters'. Each critter is a JavaScript object that has random x and y co-ordinates within the canvas, a random size, and a Boolean that indicates that the critter is alive.

   The program draws the critters onto the canvas. Each critter is drawn as a yellow square at the appropriate co-ordinates on the canvas. Note that critters might overlap with each other.

   During the game, the user must repeatedly click on the canvas. If the co-ordinates of a click fall within one or more critters, those critters die and should no longer appear on the screen.

   Complete the JavaScript program, which is shown on the next page, by giving definitions of the following functions:

   - `draw_critters`, which draws the critters onto the canvas (if they are alive); and
   - `handle_click`, which 'kills' critters that the user has clicked on.

   Together they should ensure that the user only sees live critters on the canvas. (If you need other functions, feel free to define those as well.)

   Notes:

   - You can draw on the canvas using the following from lectures:

Clearing a rectangle:	
`context.clearRect(x, y, width, height)`	
`x`	The x-coordinate of the upper-left corner of the rectangle to clear
`y`	The y-coordinate of the upper-left corner of the rectangle to clear
`width`	The width of the rectangle to clear, in pixels
`height`	The height of the rectangle to clear, in pixels
Setting the drawing colour, e.g., to yellow:	
`context.fillStyle = 'yellow';`	
Drawing a coloured rectangle:	
`context.fillRect(x, y, width, height)`	
`x`	The x-coordinate of the upper-left corner of the rectangle
`y`	The y-coordinate of the upper-left corner of the rectangle
`width`	The width of the rectangle, in pixels
`height`	The height of the rectangle, in pixels

   - You can find out the co-ordinates of a click within the canvas by using the `get_click_x` and `get_click_y` functions supplied on the next page.
   - You are not required to write any code to handle the end of the game, when all the critters are dead.

```
(function() {

    var critters = [];

    document.addEventListener('DOMContentLoaded', init, false);

    function init() {
        canvas = document.querySelector('canvas');
        context = canvas.getContext('2d');
        width = canvas.width;
        height = canvas.height;
        create_critters();
        draw_critters();
        canvas.addEventListener('click', handle_click, false);
    }

    function getRandomNumber(min, max) {
        return Math.round(Math.random() * (max - min)) + min;
    }

    function create_critters() {
        for (var i = 0; i < 10; i += 1) {
            var critterSize = getRandomNumber(2, 50);
            var critter = {
                size : critterSize,
                x : getRandomNumber(0, width - critterSize),
                y : getRandomNumber(0, height - critterSize),
                alive : true
            };
            critters.push(critter);
        }
    }

    function get_click_x(event) {
        var rect = canvas.getBoundingClientRect();
        var x = event.clientX - rect.left;
        return x;
    }

    function get_click_y(event) {
        var rect = canvas.getBoundingClientRect();
        var x = event.clientY - rect.top;
        return x;
    }


})();
```