CS210 - Assessed Coursework Exercise #2

To be completed individually

Available Marks: 30, plus 3 bonus marks

Submission deadline: 23:55 Sunday, 28th October, Week 6

How to submit: MyPlace (*submission link will be made available on Monday, 22nd October*)

General instructions:

- *You must use Unix to compile & run your programs.*
- *All your programs must comply with the **C99** standard. Make sure that you use the appropriate gcc option to achieve this. Refer to the respective lecture slides if you do not remember how to do so.*
- ***Reminder regarding Labs:** Please remember that lab assistants are NOT there to help with your Assessed Coursework Exercise (ACE). You may ask a lab assistant technical questions regarding your ACE, e.g. "how can I print an integer on the screen?", "how can I read user input?" etc., but priority will be given to students working on the Practical. Do NOT ask a lab assistant to clarify parts of your ACE. This will be done in class (Q&A) and through the class forum.*

Assessed tasks

Your task is to implement the well-known Tic-tac-toe game in C. You can check the rules here: https://en.wikipedia.org/wiki/Tic-tac-toe. Your implementation must strictly follow the specification below. You are allowed to have utility functions to support your implementation, but you are not allowed to modify the declarations of the functions described below. ***Furthermore, you are expected to collect/refine requirements via the class forum dedicated to ACE#2.***

1. Implement a C function that prompts the user to: i) select to play against another user or the computer, ii) choose 'X' or 'O', and iii) enter a nickname.

Function declaration: `void user_prompt(char *message, char *answer);`

(3 Marks)

2. Implement a C function that clears the game board.

Function declaration:

`void clear_board(`*`your game board represented as a two-dimensional array of characters`*`);`

(3 Marks)

3. Implement a C function that displays the game board.

Function declaration:

`void display_board(`*`your game board represented as a two-dimensional array of characters`*`);`

(3 Marks)

4. Implement a C function that simulates a user move.

Function declaration:

`void user_move(`*`your game board represented as a two-dimensional array of characters`*`);`

(5 Marks)

5. Implement a C function that simulates a computer move.

Function declaration:

`void computer_move(`*`your game board represented as a two-dimensional array of characters`*`);`

(5 Marks)

6. Implement a C function that detects a win.

Function declaration:

`int detect_win(`*`your game board represented as a two-dimensional array of characters`*`);`

(5 Marks)

7. Implement a `main()` C function that uses the above functions (and any additional utility functions you have implemented) to simulate the game.

(3 Marks)

Some further requirements:

1. Your program must display the game board after each move.
2. Your program must display meaningful information when a game is over.

Optional Task for a total of 3 bonus marks

Enhance your implementation in order to allow a series of games, i.e. instead of a single game. Your program must keep track of wins for each player and offer the user the option to terminate the game.

Marking Scheme

- **Completeness** (has all required functionality been implemented?), and **correctness** (does everything work as specified?)

  As specified by the marks below each exercise

  [27 Marks]

- **Commenting** (i.e. is everything commented as it should?)

  Minimum requirements:

  - Description for each function
  - Description at the top of each C file (*template is provided on MyPlace*)
  - Consistency between descriptions and implemented functionality

  [1 Mark]

- **Style**

  Minimum requirements:

  - Code nicely laid out and formatted
  - Well-chosen variable/function names
  - Source code modularity
  - Use of constants instead of literal values
  - Quality of output: new lines, values separated from text, displayed info is accurate, meaningful error messages

  [2 Marks]

- **Optional Task**

  Completeness (has all required functionality been implemented?), and **correctness** (does everything work as specified?)

  As specified by the marks below the exercise

  Partial marks available

[3 Marks]

What to submit:

A single compressed (e.g. .zip) file that contains:

1. All your source code (i.e. C files). NO executable files!
2. A Unix log script that contains a capture of your screen while working on your assessment.
   - Details of how to generate this has been given in class (*please check the respective slides*).
   - The script must contain entries of compiling and executing ALL submitted code.
   - The script must be generated using a CIS machine (NOT your personal machine) so that your username is visible.
   - The script will be used to check the originality of your work. There are no marks available for submitting the script. However, there is a **10% penalty** for those who fail to submit the script.

Penalties will be applied for late submission of assessed coursework according to the Departmental Policy.

Good luck!