**University of Wolverhampton**

**School of Engineering, Computational and Mathematical Sciences**

**4CS001 Introduction Programming and Problem Solving**

**Coursework 2**

**This assignment is worth 60% of the overall module grade**

**Introduction**:

This coursework will assess your knowledge of Python programming and computational problem solving. It also builds on many of the topics we have covered over the course of the module.

**Task**

Your task is to develop a program that play the classic game of Noughts and Crosses, or Tic-Tac-Toe.

**Overview:**

Noughts and Crosses is a simple classic game for 2 players, played on a three-by-three grid, in which players take it in turns to place their mark, either a nought O or a cross X, in a cell of the grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. For example:

Table

Description automatically generated with low confidence

To start the game, the convention is that X plays first. For the program you are going to develop, a human player plays against the computer, and the human player goes first with the cross X.

**Getting Started:**

Start by download the two Python files play\_game.py and noughtsandcrosses.py. The file play\_game.py is the main game program and it is already complete and it imports the noughtsandcrosses.py module in order to play the game. Your task is to fully complete the module noughtsandcrosses.py.

As with the previous Coursework, a number of incomplete functions are given to you and you must use these as they are. You must not change their arguments or the way they return values.

**Requirements:**

The requirement for each function is detailed in template file noughtsandcrosses.py and also explained below:

**def draw\_board(board) (2%)**

You are required to develop code to draw the noughts and crosses board, for example:

Text

Description automatically generated

**def welcome(board) (2%)**

You are required to develop code to print the welcome message and display the board by calling draw\_board(board). For example:

Graphical user interface, text, application

Description automatically generated

**def initialise\_board(board) (2%)**

You are required to develop code to set all elements of the board to one space ' '

**def get\_player\_move(board) (5%)**

You are required to ask the user for the cell to put the X in, and return row and col

A picture containing text

Description automatically generated

The player has to input a valid cell.

**def choose\_computer\_move(board) (25%)**

You are required to develop code to let the computer chose a cell to put a nought in and return row and col

**def check\_for\_win(board, mark) (5%)**

You are required to develop code to check if either the player or the computer has won return True if someone won, False otherwise

**def check\_for\_draw(board) (4%)**

You are required to develop cope to check if all cells are occupied return True if it is, False otherwise

**def play\_game(board) (25%)**

You are required to develop code to play the game.

1. Start with a call to the initialise\_board(board) function to set the board cells to all single spaces ' '
2. Then draw the board
3. Then in a loop, get the player move
4. Update and draw the board
5. Check if the player has won by calling check\_for\_win(board, mark), if so, return 1 for the score.
6. If not check for a draw by calling check\_for\_draw(board).
7. If drawn, return 0 for the score
8. If not, then call choose\_computer\_move(board) to choose a move for the computer
9. Update and draw the board
10. Check if the computer has won by calling check\_for\_win(board, mark),
11. If so, return -1 for the score
12. If not check for a draw by calling check\_for\_draw(board)
13. If drawn, return 0 for the score
14. Repeat the loop from (3)

**def menu() (5%)**

You are required to develop code to get user input of either '1', '2', '3' or 'q'. For example:

Text

Description automatically generated

**def load\_scores() (10%)**

You are required to develop code to load the leaderboard scores from the file 'leaderboard.txt'. The function has to return the scores in a Python dictionary, with the player names as key and the scores as values.

**def save\_score(score) (10%)**

You are required to develop code to ask the player for their name and then save the current score to the file 'leaderboard.txt'

**def display\_leaderboard(leaders) (5%)**

You are required to develop code to display the leaderboard scores passed in the Python dictionary parameter "leader"

**Structure and Documentation**

The structure of your code and documentation will be analysed and assessed.

This will be done using a static analysis tool called **Pylint**. This software checks the code in your program, ensures that it follows Python conventions and that all functions, classes and modules have been documented. You can read more about it here: <https://www.pylint.org/>.

Python has an official style guide named PEP8 (<https://peps.python.org/pep-0008/>), which is where most Python conventions and coding standards originate from.

Example checks that Pylint carries out to ensure that the PEP8 coding standard is followed include things such as:

* + checking line-code's length
  + checking if variable names are well-formed
  + checking if imported modules/functions are used
  + checking if variables/function parameters are used

It is a good idea to run these checks on your code at regular intervals and before submitting.

**Note:** Marks will be deducted for warning and errors detected in your code.

**Submission and Marking**

You should upload your submission to Canvas before the deadline on Canvas. If you fail to do so, you may receive a grade of 0 NS (Non-Submission).

Your work will be automatically tested using a program that tests each of the individual functions you have implemented. Therefore, it is very important that you do not alter any of the function signatures in the template and implement everything