Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CMT120
Module Title:	Fundamentals of Programming
Lecturers:	Federico Liberatore, Martin Chorley, Natasha Edwards
Assessment Title:	'Programming Challenges' [redacted]
Date Set:	17th July 2023
Submission date and Time:	7th August 2023 at 9:30AM
Return Date:	18th August 2023

This assignment is worth 40% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:
`https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf`

Submission Instructions

All coursework should be submitted via upload to Learning Central.

Description	Type	Name
Cover sheet	`.pdf` file	`[Student number].pdf`
Task 1 source file	`.py` or `.js` file	`[Student number]_1.py` or `[Student number]_1.js`
Task 2 source file	`.py` or `.js` file	`[Student number]_2.py` or `[Student number]_2.js`

Any code submitted will be run on a system equivalent to the laptops provided to the students, and must be submitted as stipulated in the instructions above. The code should run without any changes being required to the submitted code, including editing of filenames.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a deduction of 25% for the corresponding task.
Staff reserve the right to invite students to a meeting to discuss coursework submissions.

Assignment

This coursework is comprised of twp tasks which you need to complete for this assignment.

Task 1 - File System Display

Write a command line program that prints the directory and file system tree structure that originates in the current directory. The algorithm used to explore the tree must be recursive. This task can be solved either in Python or JavaScript. Sample output:

```
./
----picture1.png
----slideShow.pptx
----UniProject
--------doc1.docx
--------doc2.docx
--------old files
------------doc1.old
----personalstuff
--------holidays.xlsx
```

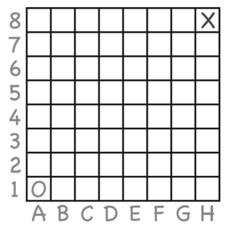
In the example, the current folder containts two files (i.e., ”picture1.png” and ”slideShow.pptx”) and two directories (i.e., ”UniProject” and ”personalstuff”). Directory ”UniProject” contains two file (i.e., ”doc1.docx” and ”doc2.docx”) and one folder (i.e., ”old files”) with one file. Directory ”personalstuff” contains only one file, i.e., ”holidays.xlsx”.

Task 2 - Amoeba Wars

Write a command line program that allows to play the game of Amoeba Wars for two human players. A description of the game is given below. This task can be solved either in Python or JavaScript.

. . . .

The game is played on a square grid. The first player is ’O’ and the other is ’X’. The game starts with an ’O’ amoeba in the bottom left corner, and an ’X’ amoeba in the top right corner (all the examples suppose a game played on an 8x8 board):

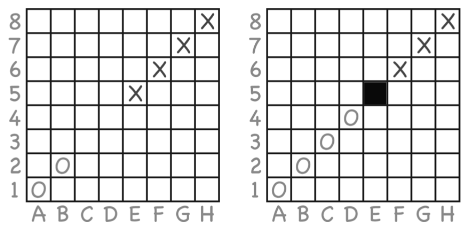


’O’ plays first and has one move. Then, the players take turns in making three consecutive moves. Each move can be to:

- Create a new amoeba, by writing the player’s symbol in an accessible empty cell, or
- Kill one of the opponent’s amoebas in an accessible cell, by shading the cell. A shaded cell is not empty and is not an amoeba.

A cell is accessible if it is next to one of the player’s live amoebas, horizontally, vertically, or diagonally.

For example, in the following opening firstly ’O’ played at B2, secondly ’X’ created three new Amoebaes (i.e., G7, F6, and E5), and finally ’O’ created two (i.e., C3 and D4) and killed one of ’X’s (i.e., E5):



A player must make all of their moves in a turn. The game ends when a player cannot make all the moves. The score of a player is the number of live amoebas they have when the game ends, plus the number of opponent's amoebas killed. The player with the highest score wins.

....

Your program should allow to play a two-players game of Amoeba War from the command line. Initially, the game asks for the size of the board. The board must be square (e.g., 6x6, 7x7, 8x8). Once the size has been entered, the initial board is drawn and the 'O' player starts. The drawing displays the board, the column and row references (i.e., A, B, C,... and 1, 2, 3,..., respectively), and the current score on top of the board, as illustrated (supposing an 8x8 board):

```
   O 01 - 01 X
-----------------
8| | | | | | | |X|
7| | | | | | | | |
6| | | | | | | | |
5| | | | | | | | |
4| | | | | | | | |
3| | | | | | | | |
2| | | | | | | | |
1|O| | | | | | | |
-----------------
 |A|B|C|D|E|F|G|H|
```

On each turn, the program informs the current player of the moves left and provides a list of feasible moves. For example, the feasible moves for player 'O' in the above board are: A2, B2, and B1. The players can make their moves one at the time by entering the coordinates of the chosen cell (e.g., A2). The program recognises invalid moves and informs the player in case of a mistake. After each move, the program draws the state of the board, informs the current player of the moves left, and provides a list of feasible moves. Finally, the program recognises when the game is over and a message is displayed to show the final score and congratulate the winner. After that, the program asks if the players want to play another game.

Learning Outcomes Assessed

- LO1: Use high-level programming languages to complete programming tasks
- LO2: Demonstrate familiarity with programming concepts, simple data-structures and algorithms
- LO3: Design and use relational databases

Criteria for assessment

Credit will be awarded against the following criteria, depending on the task.

Functionality: Does the code perform the required task correctly, accurately, and efficiently?

Code quality: Is the code elegant and well-written; simplified by the use of built-in languages features where appropriate; readable; commented? Are appropriate functions defined and written to enable reuse? Are classes defined and used effectively?

The mark breakdown for each part is given in the following.

Task 1 [Total: 50 marks]

- Functionality: Recursive definition. **12.5 marks**
- Functionality: Output as expected. **12.5 marks**
- Functionality: Program correctly displays the directory and file tree structure of the current directory. **12.5 marks**
- Code quality. **12.5 marks**

Task 2 [Total: 50 marks]

- Functionality: User interface. **8 marks**
- Functionality: Game flow. **18 marks**
- Functionality: Feasible moves. **8 marks**
- Functionality: Ending condition and score calculation. **8 marks**
- Code quality. **8 marks**

For each item evaluated, partial marks can be assigned as illustrated in the following table. The rubric applies across all of the questions, with the split in functionality, code quality, usability, presentation and content defined above.

	Distinction (70-100%)	**Merit** (60-69%)	**Pass** (50-59%)	**Fail** (0-49%)
Functionality	Fully working functionality, implemented efficiently, and achieved using a relevant programming approach.	Functionality working correctly but implemented in a non-efficient fashion.	Faulty functionality which works similarly yet not exactly as described in the question, and/or presents at least an error, and/or wrong output.	Functionality not implemented or completely faulty implementation with wrong behaviour and/or output.

Code Quality	Code is elegant, with no redundancies, well commented, perfectly modular (i.e., appropriate functions and/or classes defined).	The code complies with less than seven of the characteristics required for distinction.	The code complies with less than five of the characteristics required for distinction.	The code complies with less than tree of the characteristics required for distinction.

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on the return date via email.
The feedback from this assignment will be relevant for any future programming tasks.