1. Module number	SET08122
2. Module title	Algorithms & Data Structures
3. Module leader	Dr Simon Wells
4. Tutor with responsibility for this Assessment	Dr Simon Wells
5. Assessment	See Attached coursework description
6. Weighting	40%
7. Size and/or time limits for assessment	See Attached coursework description
8. Deadline of submission	Your attention is drawn to the penalties for late submissions **3PM on Wednesday 27th March**
9. Arrangements for submission	See Attached coursework description
10. Assessment Regulations	All assessments are subject to the University Regulations
11. The requirements for the assessment	See Attached coursework description
12. Special instructions	None
13. Return of work and feedback	You will be emailed written feedback within three working weeks of the submission deadline. However you may also receive verbal feedback during your demonstration.
14. Assessment criteria	See Attached coursework description for details. This assessment covers: **LO1:** Evaluate the performance of an algorithm based on the size of a problem input **LO2:** Examine and evaluate the algorithms that work on various data structures **LO3:** Evaluate and discuss the capabilities of data structures to applicable problems **LO4:** Demonstrate a working knowledge of a relevant data structures API **LO5:** Design, develop and evaluate data structures and algorithms

Algorithms & Data Structures SET08122
Coursework Specification

1 Task

The objective of this coursework is to demonstrate your understanding of both theory and practise in relation to the content of the Algorithms and Data Structures module. Your specific task is to implement a text-based Tic-Tac-Toe (Noughts & Crosses) game using the C programming language. The core goal of your implementation should focus on your choice of data structure and algorithms needed to implement the game. You must think carefully about which data structures and algorithms are appropriate to your solution, and record this decision making process within your report.

The minimum requirement is that you will, using your knowledge both from taught interactions and self-directed learning, identify and implement appropriate data structures to represent at least the following:

- Game board - the space in which the game is played.
- Players - representing the people interacting with the game.
- Pieces - the "noughts" and "crosses" of the title.
- Positions - which pieces have been placed in which position by which players.

These elements, together with anything else that you think is necessary so that a game can be played between two people, must be implemented as a working and playable game. The minimum requirement, a working game, is sufficient to achieve a good pass. To earn a better grade there are some additional requirements:

- Your game should record the history of play and allow earlier games to be automatically replayed from this record, i.e. the sequence of moves that the players make during a game, so that each game that is played can be recorded and re-played. Subsequent automatic replaying a record of an earlier game from the record is a part of this requirement.
- Your game should support undo, i.e. once a move is made you should be able to un-apply it, returning the game state to the immediate previous state. Your players should be able to undo moves right back to the initial game state.
- Once an undo facility is working you game can be extended to support redo of moves that have been un-done.

For every data structure and algorithm that you use in your solution, you must have evaluated and justified your choice, and documented this within your report. Ideally you will have considered alternative approaches for each aspect of the program, for example, there are multiple methods for implementing a game board, each of which has advantages and disadvantages that you should consider and report upon.

If you are aiming for a grade at upper end of the grading scale (for example 80+) and have completed the aforementioned requirements then you should consider extending the core requirements to demonstrate your self-directed learning, for example, directions that you might consider could be game boards of various sizes. Whilst the default is traditionally 3x3, larger, or differently shaped boards can lead to interesting consequences in terms of how a game is played and this might require an adjustment to how you implement your game rules. Another goal might be to make an automated player so that the computer can play against the user. In its most basic form, an automated player is merely a function that selects legal moves to play, then randomly chooses one of them and plays it. More advanced functions will support the selection of better moves.

This coursework should be fun, so use your imagination, and give your creativity a free rein. Invention and originality will be rewarded by the marking scheme. I hope you enjoy working on it.

2 Submission & Deliverables

Your coursework deliverables comprise the following:

1. Source code.
2. Report.

If you are in any doubt about any of the requirements for the coursework or any aspect of the submission procedure then please contact the module coordinator for further guidance. If, due to your circumstances, you are unable to complete your assignment on time and require an extension or something similar, then please adhere to the "Fit to sit" regulations which are available from, and detailed on, MyNapier.

2.1 Demonstration

All coursework must be demonstrated. Without a demonstration your submission will not be marked. As this class is quite large, demonstrations will be held primarily during regular lab sessions but extra sessions may be organised to give everyone an opportunity to demonstrate. All students will have the opportunity to sign-up for a demo slot at a specific time. The module coordinator will contact the class closer to the deadline to organise demo slots.

During your demo slot, your marker will try to get a good idea for what you've achieved and may ask you questions. The main goal of demos is to establish that the work that you've submitted is actually your own work.

2.2 Git Submission

Your report, source code, and a final executable that runs on a JKCC machine must be committed to the Git repository in "report", "sourcecode", and "executable" folders respectively.

- Your Git repository must be named lastname_firstname_ads
- Your repository must be pushed to a hosting service, e.g. Bitbucket, Github, or the School's git server.
- You must email the Git clone URL for your repository to s.wells@napier.ac.uk at least one week before the assignment deadline. This should be the SSH clone URL (the one that starts with either git@github or git@bitbucket). This is important and failure to do so may mean that your work cannot be accessed at the deadline of the assignment with a consequent impact upon your grade.
- You should make your repository private to avoid others being able to copy your work. However you must add the user siwells as a collaborator so that your work can be retrieved at the deadline.

2.3 Sourcecode

You must use the C programming language and your program must compile at the commandline on a JKCC machine into a single executable. Using any extra libraries, beyond the C standard library that accompany a standard installation of Visual Studio or the Gnu Compiler Collection (GCC), is not permitted.

Your submission must contain all of the sourcecode required to rebuild your executable. You should also contain clear and detailed instructions in a readme.txt file for performing that process. It is in your interests to ensure that the build process is as straightforward as possible. Your submission must contain only the files required for this submission and not your entire SET08122 folder of workbook and lab content. It is your responsibility to ensure that you have placed all of the source code necessary to build your software into your repository.

2.4 Report

Your report must be no longer than 6 pages in length (excluding appendices). Appendices may be used to include supplemental data, for example test data, screenshots, or documentation, but these must be referenced from within the main body of your report. The format of the submitted report must be PDF and should include the following sections:

Title of your report.

Introduction Describing the problem & giving an overview of features.

Design Explaining how you designed & architected your software paying particular attention to the algorithms and data structures used.

Enhancements Describing the features that you would add or improve *if you had more time*.

Critical Evaluation Explaining the features that you feel work well, or work poorly, and why you think this. You should support your evaluation with experimental results.

Personal Evaluation Reflecting on what you learned, the challenges you faced, the methods you used to overcome challenges, and how you feel you performed.

References (Optional) If you have used additional resources then these should be cited. Otherwise this section may be omitted.

If you choose to typeset your report using LaTeXthen there is a Napier report template that you can use which is available from here:

`http://github.com/edinburgh-napier/aux_latex_cw_template`

3 Important Dates & Deadlines

- Clone URL to module coordinator: At least one week before the deadline.
- Submission deadline: 3PM on Wednesday 27th March
- Demos: During the regular lab sessions on Thursday 28th March.
- Return of work: you will be emailed written feedback within three working weeks of the submission deadline. However you will also receive verbal feedback during your demonstration. Note that you will also have likely engaged with teaching staff about your assignment on many occasions during the trimester and these are all opportunities for feedback.

4 Assessment Criteria & Marking Scheme

All work must be your own work. If you have been inspired by work from elsewhere then it is worth playing safe and explicitly recording the source of any code, design ideas, or approaches that you may have adopted. Properly referencing, quoting, and citing the work of others is generally a good and safe way to do this.

This coursework is worth 40% of your overall grade for this module. The remaining 60% come from the exam. The mark breakdown for this coursework is as follows:

0-40% There are a number of ways to achieve a mark in this band, but generally you will either have failed to create a working tic-tac-toe game, omitted major functionality such as a minimum requirement, have used a wholly inappropriate and unjustified approach, failed to include a report, or the report will be wholly inadequate in justifying the decisions that you've made in your code.

40-49% To achieve a mark in this band you must have developed your own working tic-tac-toe game. This means you will have designed and implemented a playable game that meets the minimum requirements. It may be based directly on an extension of the practical work covered in class and your report must adequately describe your work but might omit sections. The quality of your implementation and associated report will dictate the position of your grade within this grade band.

50-59% A submission in this mark band indicates work of a good standard. You will have met the requirements of the previous grade band and also implemented the recording of play and replay of earlier games. You will have considered the algorithms and data structures used in your solution and recorded this within your report. Your report will be well written, include all sections, and will reference the material you have used.

60-69% To achieve a mark in this band means that you have been a little more ambitious and are achieving at a level that indicates very good work. You will have met the requirements of the previous grade band and also implemented the undo and redo features, underpinned by an appropriate choice of, and implementation of, data structures and algorithms You will have considered how to evaluate your work and may have included some results. Your report will address all the necessary sections effectively, be very well written, clearly presented, and well referenced.

70-100% A submission in this mark band will consist of a game that offers an excellent level of functionality, both in terms of the number of features and their quality of implementation. You will have met the requirements of the previous grade band and also implemented all requirements to an excellent standard. Your choices of data structures and algorithms will be both appropriate and justified in your report, taking performance and complexity into account. Your game will be playable, robust, and well designed with appropriate, and justified selections of data structures and associated algorithms. Your report will be comprehensive, include all mandated sections, be very well written and presented and will correctly reference all the material you have used. This is likely to include textbooks, online forums and tutorials and some of the suggested reading for the module. Furthermore your report will provide an excellent justification for why you've implemented your solution in the manner that you've chosen. The quality of your implementation and associated report will dictate the position of your grade within this grade band. To achieve a grade in excess of 80% you must go beyond the taught material formally covered in the module and incorporate your own self-directed reading and research. Some suggestions for direction of development at this level were made above.