MOD002702 SOFTWARE IMPLEMENTATION 2022-23 TRIMESTER 1 COURSEWORK

PORTFOLIO (15%)

During the trimester, some weekly practical classes will include a programming exercise whose solutions should be archived into a portfolio and submitted along with the main assignment at the end of the trimester. These exercises can be worked on in scheduled practical class time, and *students can support/help each other* provided that no actual code is exchanged between students, whether embedded in an email or attached as a separate file, or physically keyed in on behalf of another student. In contrast verbal discussion and communication by free text grammatical social media is encouraged. In the portfolio, each successful exercise solution is worth one mark. For example if 6 exercises are specified for assessment but only 4 are successful, you will be awarded 4/6 th's of 15% i.e 10/15 towards the final mark.

MAIN ASSIGNMENT (85%)

FIRE SIMULATION

This assignment requires you to design and implement a program that models the spread of fire in a 2D forest. Such modelling is an important aspect of research in fire safety, fire drill management, fire investigation, and with applications in many computer game scenarios. One way to look at the world in software is to view the landscape as a rectangular grid of cells. Each cell corresponds to an area in the world which can be in one of many possible states at any given time. Rules specify how a cell changes state over time based on the states of the cells around it. In this case for example a cell can either be empty, or a tree, or a burning tree. A computer simulation involving such a system is called a *cellular automaton* and is a field of study common in artificial intelligence.

Program requirements

Note that fire simulations are very well covered in textbooks, published journals and on the internet. Your implementation **must** conform to the following two requirements:

1. Language must be C#
2. Application must only be console based

These constraints should ensure substantial development of the application from first principles is required. If you do use code derived from publically available archives (e.g. internet, textbook) then that code should be referenced and it should be made as clear as you possibly can what is code you have developed and what is code you have re-used from other sources.

Code Development

1. The simulation requires you to create a simple 2D console-based world composed of a 21x21 grid of cells, where all cells contain a tree. The boundary of the forest/grid can be regarded as a firebreak; proximity to the boundary cannot cause an internal tree to catch fire. Draw the forested world using ASCII characters such as an ampersand ('&' for a tree, an 'x' for a burning tree and a blank space for an empty cell.
2. Once the simulation is started the fire is initiated (in the first instance) with a burning tree at the central cell. No user-interaction is required other than to press a key to indicate the

end of one time-period and the start of the next, or to exit the simulation. Each key press changes the state of the simulation and thus represents a time-step (actual real units of time are irrelevant). Assume that the simulation continues until no cell is on fire.

3. After each time step, re-draw the grid and prompt the user to press Enter to initiate the next time step, or another option to exit the simulation (if it helps you can think of the simulation as being a kind of turn-based game where your turn is to press 'Enter' and the computer's turn is to redraw the forest following application of the rules that progress the forest fire).
4. During each time step each cell can be in one of three states, which could be represented numerically (eg 0, 1, 2)>

   Empty – this represents either empty ground or the site of a burnt tree.

   Tree – a tree that is not burning.

   Burning – a tree that is burning.

   During each time step the following rules should be applied to each cell to determine it's new state in the next time step;

   i. If the site is empty it remains empty.
   ii. If a tree is present and none of the neighbours are burning it remains a tree.
   iii. If a tree is present and at least one of the neighbours is burning it may or may not catch fire with a probability of 50% (use a random number generator to determine this).
   iv. If the site contains a burning tree then assume it will burn down in one time step leaving an empty site.

To simplify programming for this simulation, assume the state of a diagonal cell to the northeast, southeast, southwest, or northwest does <u>not</u> have an impact on a current cell's value at the next iteration. Therefore a cell's value at the next timestep depends on the cell's current state and the values of its neighbours to the north, east, south, and west. Note In very general terms the coding solution should firstly set up the forest environment and then secondly implement the time steps and forest updates via a loop structure. Note that your solution must be object-oriented as far as possible. Some suggestions (not exhaustive or even necessarily a requirement) are;

- Develop a class Cell class whose objects fill a 2D array called **forest**, with appropriate private attributes and public methods.
- Develop a class Grid, with a static 2D char array of ascii symbols **map** reflecting the current visual state of the forest used for displaying to screen. A static method **spread()** could process each Cell object in turn – by passing the state of the current cell and the states of it's neighbours, and return the new (ie next) state.
- Another static method could be **applySpread()** that takes at least one argument (the **forest**) and updates the forest grid and associated **map**. One approach is to have an empty copy of the grid used every turn; that is, apply the update rules in turn to each cell and then add the updated cell to the new location in the copy of the grid *not the original*. When all the cells have been updated in the copy, that copy becomes the new 'original' (which can be drawn to screen) and the old grid is discarded.

<u>Enhancements</u>

The complexity and available credit for this assignment is very dependent on which requirements you implement and how they are implemented. Aim for a basic implementation in the first instance. However to maximise your marks you are encouraged to explore techniques (own ideas, literature research) to further extend/enhance the simulation. For example the most intuitive way to code a solution is to create a 2D array of Cell objects along with a 2D char array for display purposes as described above. However

there may be more memory efficient ways of utilising an underlying data structure such as a linked list of tree objects (whether in a 'burning' or 'unburning' state) where each tree object contains a reference to a map of the whole forest, and the list shrinks or expands according to the number of trees at any one time. Another example is to introduce the effect of environment on how the fire might spread (e.g. wind speed and direction, ground moisture etc) which might influence the probability of a tree catching fire.

Assessment of Fire Simulation assignment

As is usually the case, there will be a balance between difficulty and what you can get done in the time you have available. Optionally, you may demo your program to the tutor *once* at some point in the last three weeks of the trimester, either to get advice on subsequent development or to better communicate the functionality of the program to the tutor and hence aid the marking process. This demo session if used will not be marked. Even if a program lacks a requirement, or even does not compile (!) you will still get credit based on what you have done including the documentation. The main assignment mark will be an amalgamation of three equally weighted components (see mark/feedback sheet below for details. *Note that even if your main assignment program is not fully functional (or even will not compile) you can still get credit for the program design, coding and evaluation.* Feedback will be available via e-vision and by the local system in use (e.g. Canavas, email).

Coursework submission

Structure your submission with a folder named after your sid e.g. 'sid1234567' This folder should itself contain two folders, one called 'Portfolio' and the other called 'ForestSimulator' or similar. The portfolio should contain one folder per assessed exercise, each containing at least two files, i.e. the C# source code (.cs) and the C# executable (.exe). The main assignment folder should contain;

1. Source (.cs) and compiled code (.exe) files, and any other relevant files to run the program if applicable.
2. A pdf or word file document report to include>
   a) 400 word evaluation of the simulator (for example, rationale for choice of classes and/or data structure(s); discussion on strengths and limitations of the implementation).
   b) A class diagram of your implementation.
   c) A Test Plan describing five tests of your choice and a table of five Test Results.
   d) A copy of the entire source code content reproduced in the word/pdf file as Appendix 1, suitably formatted.
   e) Instructions for running the program in an Appendix 2.

The submission, 'sid1234567' should then be compressed into a single zip file e.g. 'sid1234567.zip' and submitted by 2pm on **Friday 9th December 2022**. To minimize the size of the file submitted avoid submitting Visual Studio project files (i.e only .cs and .exe program files should be submitted, besides the word doc/pdf file constituting the report.

Reassessment

If you are unfortunate enough to require a reassessment (normally as second attempt but possibly as a first attempt), you should re-submit the Forest Fire Simulator assignment by the standard submission date for Trimester 1 reassessments and your mark out of 100% will be calculated on this new submission only (usually the system will cap this at 40%). Note there will be NO portfolio component to submit or considered irrespective of how well they were

undertaken at the first attempt. You are encouraged to explore a different approach to achieving a solution to the main assignment compared to the first attempt.

MOD002702 Software Implementation Coursework feedback sheet 2022-23 Sem1

	In-class exercises (not used in reassessment)	Main Assignment							Overall 010 mark	Overall module result
Student	**/15**	**Functionality**	**Object oriented approach and design**	**Documentation and presentation**	**Comment**	**%**	**Grade**	**/85**	**%**	**Grade**
	Weighted 0.15. Eg if 5 exercises are successfully demo'd out of 9 then the mark out of 15 is> 5/9 of 15 = 8.3	A: Includes user-configurable environmental enhancements. B: Time steps accurately reflect rules for fire spread. C: Basic simulation implemented with reliable execution. Rules not properly implemented. D: Unclear interface, clearly wrong output. F: Run time error or does not compile.	A: Uses purposeful 3+ user-written classes, accurate class diagram, restricted scope. B: Develops instance/static methods, scope not minimized. C: Uses standard library resources for data structure processing. D: Procedural not OO approach. F: Non-OO approach, no methods.	A: Excellent evaluation. B: Evaluates success and limitations of approach, includes appropriate test plan and results. C: Clearly written, clear code layout, class diagram of sufficient detail. D: Insufficient detail, poorly written. F: Substantial elements missing.	Personalized comments specific to student submission.	Each component is assigned a grade (F, D, C, B, A) and each maps to a mark (0-100) from which the average is taken.	Main assignment grade	Weighted 0.85	Overall weighted mark	F (<40); D (40-49); C (50-59); B (60 - 69); A (70+)
Sid Number	**Mark**	**Grade**	**Grade**	**Grade**	**Comment**	**Mark**	**Grade**	**Mark**	**Final Mark**	**Final Grade**