University of Westminster
School of Computer Science and Engineering

6SENG005W	Formal Methods – Coursework (2024/25)
Module leader	Klaus Draeger
Unit	Coursework
Weighting:	50%
Qualifying mark	30%
Description	Developing a B specification of a simplified *Battleships* board game.
Learning Outcomes covered in this assignment:	This assignment contributes towards the following Learning Outcomes (LOs): LO1, LO2, LO3, LO4
Handed Out:	October 2024
Due Date	13:00, Wednesday, 11th December 2024
Expected deliverables	**A specification using the B language (.mch format)**
Method of Submission:	Electronic submission on Blackboard.
Type of Feedback and Due Date:	Verbal feedback in tutorial(s) before the assessment is submitted. Sample answers of the assessment after 15 working days (3 weeks). Written feedback and marks 15 working days (3 weeks) after the submission deadline. **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description

This coursework requires you to develop a B specification of a version of the *Battleships* board game. This system is to be developed using the B tools Atelier B and ProB.

Battleships is a strategy type guessing game for two players. It is played using two 10 x 10 grids, one for each player. The game starts by each player placing the warships of their fleet on squares in their own grid. The locations of the fleets are concealed from the other player. Players take alternate turns "*shooting*" at the other player's ships in an attempt to sink them. The aim of the game is to win by destroying your opponent's fleet of ships before they destroy yours. See https://en.wikipedia.org/wiki/Battleship_(game).

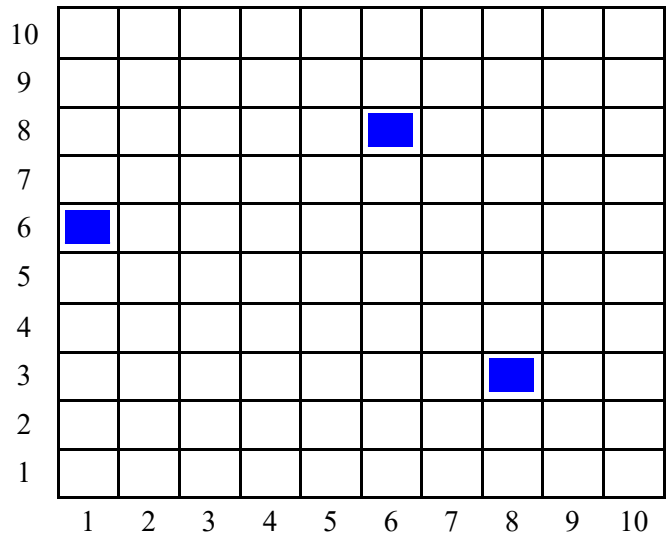

Figure 1. Example of a Player's Battleships Board with 3 ships in positions (1,6), (6,8), (8,3)

For example, if Player 1 has their fleet positioned as in Figure 1, then if Player 2 *"shoots"* at square (8,3), then they score a *"hit"* and sink the ship in that position. This means the ship is removed from the board, leaving Player 1 with just the ones in squares (1,6) and (6,8). If on Player 2's next turn they "*shoot*" at square (4,4) then it is a *"miss"* and has no effect on Player 1's ships.

The *winner* of the game is the Player who sinks all of the other Player's ships before all of their own ships are sunk.

1.1 The Battleships Game Constraints and Requirements

For the purposes of this coursework, the Battleships game used is a simplified version of the normal game:

- A grid size of 10 x 10 squares, with the grid square (6,8) representing the square at column 6 & row 8.
- Each player has 3 warships in their fleet.
- Each warship occupies a single grid square on the grid.
- The warships must be on different grid squares, i.e. at most 1 warship per square.

1.2 Playing the Battleships Game

The Battleships game is played as follows:

1. The game starts by each player placing their 3 ships on their own grid, each player's ships in different grid squares.
2. Player 1 has the first turn.
3. Player 1 shoots at Player 2's ships, by selecting a target square on Player 2's grid. If there is a ship in the target square it is a hit and the ship is deleted from Player 2's fleet, otherwise it is a miss and there is no change to Player 2's fleet.
4. It is then Player 2's turn.
5. Player 2 shoots at Player 1's ships, by selecting a target square on Player 1's grid. Registering a hit or miss, as above.
6. The players continue to take turns until all of one player's ships have been sunk, at this point the other player is the winner.

2. The Battleships Game B Specification

2.1 Specification Components

Your B specification should include the following elements.

(a) Sets and Constants
Any sets and constants that are required to define the data and state of the Battleships game system, and their properties.

Includes at least the set of players, dimensions of their grids, and number of battleships.

(b) System State
The state variables required to represent the data for the Battleships game system. Including the state invariant and initialisation.

Includes at least both players' grids, the current game state (for example, it could be ongoing, won by either player, or one or both players may still need to deploy their fleet), and whose turn it is. Check the operations to figure out what other information is needed.

(c) Operations
The Battleships game operations are:

`report <-- deployFleet(player, positions)`

> The specified player places their 3 ships at the given positions on their own grid. If the ships are all placed in valid grid positions, then the operation reports *success*, otherwise it reports an *error message* indicating what the error was.

`report <-- playerShoots(target)`

> The player whose turn it is shoots at the target square on the other player's grid. If it contains a ship, then this ship is deleted, and the operation reports a *hit.* If this was their last ship the game is over, and the current player wins the game; otherwise, it is the other player's turn. If there is no ship in the target square the operation reports a *miss.* If the target square is not in the grid, it reports an error message.

For each operation make sure to use suitable preconditions (e.g. players should not start shooting before both fleets have been deployed).

(d) Enquiry Operations

You must also specify the following enquiry operations, that all output information about the state of the game:

`shipsquares <-- shipLocations(player)`

Outputs the location of all the ships left in the player's fleet.

`shipCounts <-- shipsLeft`

Outputs the numbers of ships left in both players' fleets.

`shotCount <-- shotsTaken(player)`

Outputs the number of shots taken by the player.

`report <-- gameStatus`

Gives information about the game's state (e.g. ongoing, player 1 win, both players still need to deploy, etc)

2.2 General Requirements

The B specification should use the appropriate features to define the data & operations in any machines that you define. The overall structure of the B specification can be either:

- A single B machine that contains all the state information and operations, etc.
- A collection of B machines that together contain all the state information and operations, etc. The appropriate B structuring features must be used to combine the collection of B machines.

3. Blackboard Submission

Submit your B machine(s) on blackboard. They should be in mch format in plain ascii files, either just a single file or zipped if more than one.

Coursework marking scheme:

Criterion and range	Indicative mark	Comments
Sets and Constants (0-15 marks)	11-15	All necessary sets and constants have been defined, including their properties. Types of constants are suitable.
	6-10	Sets, constants and properties are incomplete or not entirely correct.
	0-5	Major parts are missing or wrong.
Variables (0-15 marks)	11-15	All necessary variables have been defined, including their invariants and initialisation. Types are suitable.
	6-10	Variables, invariants and initialisation are incomplete or not entirely correct.
	0-5	Major parts are missing or wrong.
deployFleet operation (0-15 marks)	11-15	Operation is defined correctly and does what is expected.
	6-10	Operation is mostly correct.
	0-5	Operation is missing or inherently incorrect.
playerShoots operation (0-15 marks)	11-15	Operations are defined correctly and does what is expected.
	6-10	Operations are mostly correct.
	0-5	Operations are missing or inherently incorrect.
shipLocations operation (0-10 marks)	8-10	Operation is defined correctly and does what is expected.
	4-7	Operation is mostly correct.
	0-3	Operation is missing or inherently incorrect.
shipsLeft operation (0-10 marks)	8-10	Operation is defined correctly and does what is expected.
	4-7	Operation is mostly correct.
	0-3	Operation is missing or inherently incorrect.
shotsTaken operation (0-10 marks)	8-10	Operation is defined correctly and does what is expected.
	4-7	Operation is mostly correct.
	0-3	Operation is missing or inherently incorrect.
gameStatus operation (0-10 marks)	8-10	Operation is defined correctly and does what is expected.
	4-7	Operation is mostly correct.
	0-3	Operation is missing or inherently incorrect.