Declaration

Course code and name:	B37VB Praxis (programming)
Type of assessment:	**Individual** *(delete as appropriate)*
Coursework Title:	Software game project
Student Name:	Mohammad AL Bitar
Student ID Number:	H00457814

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Mohammad AL Bitar*

Date: *18/02/2025*

Implementation of a Battleship Game in C

Introduction

This report outlines the design and development of a console-based Battleship game programmed in C. The game utilizes a 10x10 grid to represent the ocean battlefield. Ships are randomly placed for the player and Ai opponent on their respective grids. The ships have different sizes and are represented internally using structures. The player and AI take turns attacking specific grid coordinates to sink each other's fleet. The game visually displays the grid state via the terminal, with symbols to represent water, ships, hits, and misses.

Implementation Details

Grid Initialization

At the start of the game, both the player and AI grids are initialized to represent open water. Each grid cell is marked with a water symbol. This creates a consistent and clear interface for further actions such as placing ships or registering hits. The program includes a method for displaying the grid on the terminal, with an option to either show or hide the locations of the ships. This helps in managing the game logic for both player view and AI logic purposes. Ships are placed randomly in either vertical or horizontal orientation. The code ensures that no ship extends beyond the grid boundary and that ships do not overlap. This is achieved using randomization combined with boundary checking and verification of clear space before placement. (Note that the AI implementation was taken from inspiration from a couple sources I found and will link them in the references section).

Game Mechanics

This type of game is a turn-based game, alternating between the player and the AI. Each turn involves selecting coordinates to target. If the target contains part of a ship, it is marked as a hit; otherwise, it is marked as a miss. The game continues until one side has all their ships sunk. This logic is implemented through conditional checks and loop structures. The AI is designed with basic strategic capabilities. Upon landing a successful hit, the AI enters a targeting mode that focuses on adjacent grid cells to complete the destruction of the ship. This mode is managed through a structure that stores the last known hit coordinates and a flag indicating targeting behavior. Otherwise, the AI attacks randomly across the grid.

Code Design Considerations

The program is structured using modular design principles. Each major functionality—such as initializing the grid, placing ships, handling attacks, and controlling AI behavior—is encapsulated in its own function. This modularity improves readability, simplifies debugging,

and promotes code reuse. Comments are used throughout the codebase to enhance clarity, and logical naming conventions make the source code easy to follow.

Conclusion

The Battleship game in C successfully demonstrates key concepts in programming logic, structured data management, and AI design. It provides a practical learning platform for engineering students to apply C programming in a meaningful context. The project can be expanded in the future to include graphical interfaces, multiplayer functionality, or improved AI strategies for a more challenging experience.

References

for ai logic: https://github.com/gabrieledcjr/Battleship/blob/master/Battleship/battleship.c

https://github.com/kyliegun/Battleship-Game-In-C/blob/main/opponentAI.c