PROJECT BATTLESHIP

GROUP NAME	**C Sailors**
GITHUB URL:	**HTTPS://GITHUB.COM/SULEIMAN50/PROJECT-BATTLESHIP**

HIGH LEVEL DESCRPIPTION

We implemented a bot of three difficulties. In general, the difficulty stems from choosing the box to attack or whether to use a special move or not. The bot works on targeting mode which sinks a ship and a hunter mode which finds a ship. The game relies on a value (we call probability) on each box where the box with the highest value probably has a ship. This method is used for hard and medium mode with the values calculated differently. The easy mode is based on a a checkers pattern for hunting and breadth first search for targeting. The usage of special moves varies between levels.

STRATEGIES USED

1- SLIDING WINDOW TECHNIQUE FOR GENERATING THE PROBABILITY GRID
2- MONTE CARLO SIMULATION FOR A MORE ACCURATE PROBABILITY GRID
3- OUTPUT MANIPULATION FOR THE ILLUSION OF ANIMATED TEXT
4- MULTIPLE STRUCTS FOR IMPORTANT DATA

ISSUES	RESOLUTIONS
1. Figuring out the most optimal to choose the boxes to fire at.	1. After research and deep thinking, we implemented 3 methods, one based on Monte Carlo Simulation, one based on sliding window, and one based on BFS.
2. Determining when to use special moves optimally.	2. Each special move had an approach to when the bot uses it, and each difficulty had a randomized slider affecting the probability of using the move.
3. The game wasn't visually pleasing and seemed abstract.	3. We added a Menu, a list of names for the bots, interactive gaming, coloring , and animations for the dialogues and the grid.
4. Scanner overloading ruined the user experience	4. We read documentations about the scanner in C, so we used the function getChar() to medicate the issue.

LIMITATIONS

We implemented multi-threading initially. However, due to the several libraries it requires, it could cause compilation errors if some libraries aren't downloaded in the compiler. Note that over 10,000 samples, we had a benefit of 0.6 seconds, which isn't very big in our case, but over a larger input, we could get a benefit of 12 seconds.

ASSUMPTIONS

1. ☐ **In most cases, we assumed and covered worst cases**
2. ☐ **Monte Carlo's random samples are picked uniformly**