Battle Boats writeup

We were tasked to create a C# program/game which mimicked the popular board game of battleships.
Here is:

01. Success Criteria
02. Design Document
03. My Code
04. Evaluation

Success Criteria

This is what we were asked to do. I have ticked and marked the ones I have done.

Main Tasks

Develop the section of the program responsible for presenting a menu to the user, giving them the option to start a new game, resume a game, read the instructions, or quit the game.

- [x] ~~Develop this section of the program in a manner that, when the user selects "play a new game," they will be shown a blank fleet grid (figure 3). The program should:~~
  - [x] ~~Prompt the user to enter coordinates for each boat.~~
  - [x] ~~Verify if a boat has already been placed in the specified location.~~
  - [x] ~~Display each boat on the fleet grid after each entry.~~
  - [ ] ~~Only allow the user to enter five boat locations.~~
    - ~~For added enjoyment, I have decided to allow the user to enter anywhere from 1 - 32.~~
- [x] ~~Develop the section of the program that randomly selects five unique locations for the computer's fleet. These locations should not be revealed to the user. Please note that, in the final version of the game, these locations should not be displayed.~~
- [x] ~~Develop the section of the program that displays a blank target tracker (figure 4) for the user.~~
- [x] ~~Develop the section of the program that allows the user to take their turn. The program should:~~

- [x] ~~Prompt the user for the target coordinates. Please note that the program should not allow the user to select the same coordinates twice.~~
- [x] ~~Check if the target is a hit or a miss.~~
- [x] ~~If the target is a hit, display an X on the target tracker.~~
- [x] ~~If the target is a miss, display an O on the target tracker.~~

- [x] ~~Develop the part of the program that allows the computer to take its turn. The program should:~~
  - [x] ~~Randomly generate target coordinates. NOTE: The program should not allow the computer to select the same location twice.~~
  - [x] ~~Display the coordinates to the player.~~
  - [x] ~~Check if the target is a hit or a miss~~
  - [x] ~~If the target is a hit then an H should replace the B (figure 5) on the fleet grid.~~
    - ~~Again I changed some of the symbols but yes :)~~
    - ~~It is now:~~
      - ~~~ for blank~~
      - ~~X for hit~~
      - ~~O for a ship~~

- [x] ~~Develop the section of the program that keeps the game going until there is a clear winner. A game is considered won when a player successfully sinks all of their opponent's battle boats. Make sure to display the winner of the game to the player.~~

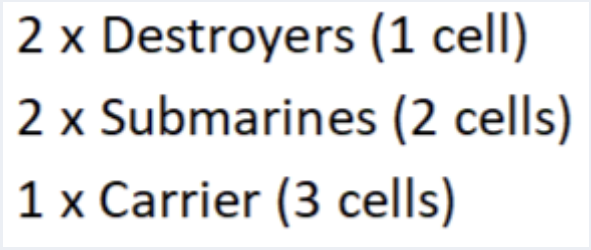
Challenge Tasks

- [ ] Develop the part of the program that saves the progress of the game externally. Progress should be saved after each turn. If the player closes the game window (stops the execution) then all progress should be saved.
- [ ] Develop the part of the program in such a way that when ‘resume a game’ is selected, the player is presented with their progress from their previous game. The game should continue from this point.

Further Challenge Tasks

- [ ] Extend the game so that the user can play a version of the game that uses a variety of boat sizes.

- [ ] The boats will come in three categories:

  - [ ] The boats can be placed horizontally or vertically.
- [ ] To allow for varying boat sizes. Develop the program so that it now checks for a hit, a miss and a sunken boat. The boat will only sink if all parts of the boat have been hit.

Design Document

01. **The main game loop:**
    The game is structured as a loop that continues as long as both the player and the CPU have remaining ships (`playerShipsLeft` and `cpuShipsLeft` are greater than zero).
02. **The grid display:**
    The console is cleared to display the player's ships and the opponent's map. The `displayGrid` function is used to show the grids in the console.
    It's also cleared every turn just for clarity's sake.
03. **The player's turn:**
    - The player is prompted to enter coordinates (in the format like G2) to target the opponent's ships.
    - The entered coordinates are validated, and if valid, the program checks the opponent's grid at those coordinates:
      - If there's a ship (`'O'`), the player scores a hit, and the opponent's ship is marked as sunk (`'X'`).
      - If the cell is empty (`'~'`), it's a miss, and the cell is marked with an asterisk (`'*'`).
      - If the player has already fired at those coordinates, it lets them enter another coordinate.
      - If the entered coordinates are invalid (syntax - wise), an error message is shown.
04. **The CPU's Turn:**

- The CPU generates random coordinates and checks if it has already fired at those coordinates. If it has, it randomizes new coordinates until it finds an unfired cell.
- The program then checks the player's grid at those coordinates:

- If there's a ship (`'O'`), the CPU scores a hit, and the player's ship is marked as sunk (`'X'`).
- If the cell is empty (`'~'`), it's a miss, and the cell is marked with an asterisk (`'*'`).

05. **Game Status:**
    After each turn, the current status of each player's fleet is displayed.
    Could be the same as #1 grid display to be fair
06. **End of Game:**
    - If the player's fleet is destroyed (`playerShipsLeft == 0`), a losing message is displayed, and the option for a rematch is offered.
    - If the CPU's fleet is destroyed, a winning message is displayed, and the option for a rematch is offered.
07. **Rematch:**
    The `Rematch` function is called to prompt the user for a rematch, providing a continuation of the game based on the winner's result.

My Code

Program.cs on Github.com:

https://github.com/pixeljammed/hrsfc-programs/blob/main/program.cs

Image of code

(warning it's massive - almost 500 lines)

```
{}  using System;
    using System.Drawing;
    using System.Globalization;
    using System.Transactions;

    namespace Battleships;

    class Program
    {
        static void Main(string[] args)
        {
            Task.Delay(1000);
            ShowMenu();
        }

        static void cooltext()
        {
            Typewrite( message: "milo tek - 2023");
            Console.WriteLine("         __                    __      __      __                    __        __                    ");
            Console.WriteLine("        /  |                  /  |    /  |    /  |                  /  |      /  |                   ");
            Console.WriteLine("        $$ |____    ______   _$$ |_  _$$ |_   $$ |  ______    ______ $$ |____  $$/   ______    ______ ");
            Console.WriteLine("...      $$      \\  /      \\ / $$   |/ $$   |  $$ | /      \\  /      \\ |$$      \\ /  | /      \\  /      |");
            Console.WriteLine("        $$$$$$$  | $$$$$$  |$$$$$$/ $$$$$$/   $$ |/$$$$$$  |/$$$$$$$/ $$$$$$  |$$ |/$$$$$$  |/$$$$$$$/ ");
            Console.WriteLine("...      $$ |  $$ | /    $$ |  $$ | __ $$ | __ $$ |$$    $$ |$$      \\ $$ |  $$ |$$ |$$ |  $$ |$$      \\ ");
            Console.WriteLine("        $$ |__$$ |/$$$$$$$ |  $$ |/  |$$ |/  |$$ |$$$$$$$$/  $$$$$$  |$$ |  $$ |$$ |$$ |__$$ | $$$$$$  |");
            Console.WriteLine("        $$    $$/ $$    $$ |  $$  $$/ $$  $$/ $$ |$$       |/     $$/ $$ |  $$ |$$ |$$    $$/ /     $$/ ");
            Console.WriteLine("        $$$$$$$/   $$$$$$$/    $$$$/   $$$$/  $$/  $$$$$$$/ $$$$$$$/  $$/   $$/ $$/ $$$$$$$/  $$$$$$$/  ");
            Console.WriteLine("                                                                                    $$ |                ");
            Console.WriteLine("                                                                                    $$ |                ");
            Console.WriteLine("                                                                                    $$/                 ");
        }









        ///  GAME FUNCTION ///

        static void game()
            {
                // cool ascii art :)
                Console.Clear();
                cooltext();




                // get user to enter ship #

                int shipsCount = 0;
                do
                {
                    Typewrite( message: "Enter number of ships per person [8~ RECOMMENDED]: ");

                    string input = Console.ReadLine();
                    if (int.TryParse(input, out shipsCount))
                    {
                        if (shipsCount > 32 || shipsCount <= 0)
                        {
                            Console.WriteLine("Invalid input. Please enter a number between 1 and 32 [8~ RECOMMENDED].");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Invalid input. Please enter a valid integer.");
                    }
                } while (shipsCount > 32 || shipsCount <= 0);




                /// INITIALIZATION ///
```

```
73
74          char[,] playerGrid = createGrid();
75          char[,] playerDisplay = createGrid();
76          char[,] cpuGrid = createGrid();
77          randomizeGrid(cpuGrid, shipsCount);
78
79          int playerShipsLeft = shipsCount;
80          int cpuShipsLeft = shipsCount;
81
82          Random rnd = new Random();
83
84          int cpuX;
85          int cpuY;
86
87
88          // let user decide to populate manually or automatically
89          Typewrite( message: "Aye... would ye like to have your ships placed at random or for you to put in yourself? \n" +
90                    "1 - randomly \n" +
91                    "2 - manually \n \n");
92
93          int choice;
94          while (!int.TryParse( s: Console.ReadLine(), out choice) || choice < 1 || choice > 2)
95          {
96              Typewrite( message: "Invalid input. Please enter either 1 for random or 2 for manual. \n \n");
97          }
98
99          switch (choice)
100         {
101             case 1:
102                 Console.WriteLine("You selected RANDOM!");
103                 randomizeGrid(playerGrid,shipsCount);
104                 break;
105             case 2:
106                 Console.WriteLine("You selected MANUAL ENTERING");
107                 populateGrid(playerGrid, shipsCount);
108                 break;
109         }
110
111
112
113
114
115         /// GAME LOOP ///
116
117         while (playerShipsLeft > 0 && cpuShipsLeft > 0)
118         {
119             // Clean up display, show da grids
120             Console.Clear();
121
122             Console.WriteLine("Your ships:");
123             displayGrid(playerGrid);
124
125             Console.Write("\n \n");
126
127             Console.WriteLine("Opponent map:");
128             displayGrid(playerDisplay);
129
130             Console.WriteLine("\n \n YOUR TURN CAPTAIN! Enter coords: (ex: B4): \n \n");
131             var fireCoords :(int,int) = formatToCoordinates( og: Console.ReadLine());
132             int fireX = fireCoords.Item2;
133             int fireY = fireCoords.Item1;
134
135             if (fireX >= 1 && fireX <= 8 && fireY >= 1 && fireY <= 8) // check if within grid :p
136             {
137                 if (cpuGrid[fireX, fireY] == 'O')
138                 {
139                     Typewrite( message: "Hit! You sank a ship!");
140                     playerDisplay[fireX, fireY] = 'X';
141                     cpuShipsLeft--;
142                 }
143                 else if (cpuGrid[fireX, fireY] == '~')
144                 {
```

```
                Typewrite( message: "Miss! Try again.");
                playerDisplay[fireX, fireY] = '*';
            }
            else
            {
                Console.WriteLine("You've already fired at these coordinates. Try again.");
            }
        }
        else
        {
            Console.WriteLine("Invalid coordinates. Try again.");
        }

        /// CPU CODE ///
        //
        // There's no reason for the CPU to actually have any code, honestly
        // it could just be {shipCount}/64 chance of randomly hitting one of your ships
        // BUT ALAS. we ball...

        // tension builder lmao
        for (int i = 0; i < rnd.Next(7,15); i++)
        {
            Console.Write('.');
            Thread.Sleep( millisecondsTimeout: 500);
        }

        // Actual "CPU" code

        cpuX = rnd.Next(1, 9);
        cpuY = rnd.Next(1, 9);

        while (playerGrid[cpuX, cpuY] == '~' )
        {
            // regenerate random coordinates if the CPU has already fired there
            cpuX = rnd.Next(1, 9);
            cpuY = rnd.Next(1, 9);
        }

        switch (playerGrid[cpuX, cpuY])
        {
            case 'O':
                Typewrite( message: "The opposing force has managed to sink one of our ships.");
                playerGrid[cpuX, cpuY] = 'X';
                cpuShipsLeft--;
                break;

            case '~':
                Typewrite( message: "The opponents missed any of our ships this time. Phew!");
                playerGrid[cpuX, cpuY] = '*';
                break;
        }




        Console.WriteLine("\nYour fleet: " + playerShipsLeft + " | CPU's fleet: " + cpuShipsLeft);
    }


    /// LOSE OR WIN ///

    if (playerShipsLeft == 0)
    {
        Typewrite( message: "You LOST... \n" +
                            "Your fleet is doomed. Hundreds of men have been lost, thanks to your incompetence \n" +
                            "Now it is you who must decide... \n" +
                            "...rematch this twisted, evil being, hellbent on destroying our fleet? \n");
        Rematch(); //offer rematch to loser
    }
    else
    {
```

```
            Typewrite( message: "Well done solider. You have succeeded in your mission, and you live another day. \n" +
                                "Dare you risk playing another manic game against this ruthless artificial intelligence...? \n" +
                                "YES OR NO. \n");
            Rematch(); //offer another game to winner :D
        }

        Console.ReadKey();
    }
```

```
    // GRID FUNCTIONS

    static void displayGrid(char[,] grid)
    {
        for (int row = 0; row < 9; row++)
        {
            Console.Write("\n");
            Console.WriteLine(new String( c: '-', count: grid.GetLength( dimension: 0) * 2));
            for (int column = 0; column < 9; column++)
            {
                Console.Write("|");
                Console.Write(grid[row, column]);
            }
        }
    }

    static void randomizeGrid(char[,] grid, int shipsCount)
    {
        Random rnd = new Random();
        for (int count = 0; count < shipsCount; count++)
        {
            int posX = rnd.Next(2, 9);
            int posY = rnd.Next(2, 9);
            grid[posX, posY] = 'O';
        }
    }

    // COORDINATE NONSENSE FUNCTIONS

    static char[,] createGrid()
    {
        string alph = "ABCDEFGH";
        string nums = "12345678";

        // create empty grid
        char[,] grid = new char[9, 9];

        // fill grid with ~
        for (int row = 0; row < 9; row++)
```

```
        {
            for (int column = 0; column < 9; column++)
            {
                grid[row, column] = '~';
            }
        }

        // make the row A-H
        for (int pee = 0; pee < 8; pee++)
        {
            grid[0, pee+1] = Convert.ToChar(alph.Substring( startIndex: pee, length: 1));
        }

        // make the column 1-8
        for (int poo = 0; poo < 8; poo++)
        {
            grid[poo + 1, 0] = Convert.ToChar(nums.Substring( startIndex: poo,  length: 1));
        }

        return grid;
    }
    static void populateGrid(char[,] grid, int shipsCount)
    {
        for (int count = 0; count < shipsCount; count++)
        {
            Console.Clear();
            displayGrid(grid);

            Console.Write("\n \n");
            Console.WriteLine("Enter coordinate to place boat (in A1 format, EX: B4, F7): ");

            bool validInput = false;

            while (!validInput)
            {
                var input :string? = Console.ReadLine();

                try
                {
                    var coords :(int,int) = formatToCoordinates(input); // returns a tuple
                    int X = coords.Item2;
                    int Y = coords.Item1;

                    if ((X >= 1 && X <= 8) && (Y >= 1 && Y <= 8) && grid[X, Y] != 'O')
                    {
                        grid[X, Y] = 'O'; // place boat!
                        validInput = true;
                    }
                    else
                    {
                        Console.WriteLine("Invalid coordinates or already placed a boat there. Try again!");
                    }
                }
                catch (Exception)
                {
                    Console.WriteLine("Invalid input. Please enter coordinates in the correct format (e.g., B2).");
                }
            }
        }
    }

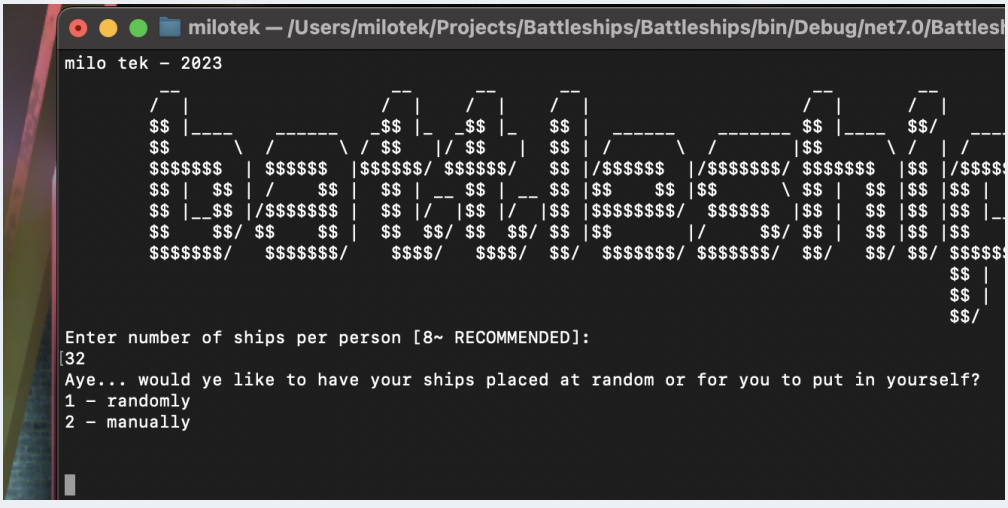

    static (int,int) formatToCoordinates(string og) // converts input like "C2" to "3,3"
    {
        og = og.ToUpper(); //fix lowercase
        int letter = charToDigit( character: Convert.ToChar(og.Substring( startIndex: 0,  length: 1)));
        int number = Convert.ToInt32(og.Substring( startIndex: 1,  length: 1));
        return (letter, number);
    }

    static int charToDigit(char character) // convert from character to digit - for above function
    {
```

```
361         return character - 64;
362     }
363
364
365     // MISC FUNCTIONS
366
367     static void Typewrite(string message)
368     {
369         for (int i = 0; i < message.Length; i++)
370         {
371             Console.Write(message[i]);
372             System.Threading.Thread.Sleep( millisecondsTimeout: 60);
373         }
374         Console.Write("\n");
375
376     }
377
378     static void Rematch()
379     {
380         string answer = Console.ReadLine();
381         if (answer.ToUpper() == "YES")
382         {
383             game();
384         }
385         else
386         {
387             return;
388         }
389     }
390
391
392
393
394
395     /// MENU STUFF ///
396
397     static void ShowMenu()
398     {
399         Console.Clear();
400
401         cooltext();
402
403         Console.WriteLine("ENTER OPTION:");
404         Console.WriteLine("1. Play Battleboats");
405         Console.WriteLine("2. Resume a game [unimplemented]");
406         Console.WriteLine("3. Read the instructions");
407         Console.WriteLine("4. Exit");
408         Console.Write("Enter number 1-4: \n \n");
409
410         int choice;
411
412         while (!int.TryParse( s: Console.ReadLine(), out choice) || choice < 1 || choice > 4)
413         {
414             Console.WriteLine("Invalid input. Please enter a number between 1 and 4.");
415         }
416
417         // Process the user's choice
418         switch (choice)
419         {
420             case 1:
421                 game();
422                 break;
423             case 2:
424                 Typewrite( message: "it dont work... mens...");
425                 ShowMenu();
426                 break;
427             case 3:
428                 Instructions();
429                 break;
430             case 4:
431                 Typewrite( message: "Bye bye!");
432                 break;
```

```
433         }
434     }
435
436     static void Instructions()
437     {
438         Console.Clear();
439         // i wrote this all in obsidian lmao
440
441         Console.WriteLine("\nAhoy there, matey! Ready to try your luck to survive out on the sea? Don't worry, it's easy as pie :^)\n\n1. **Plot
442         Console.WriteLine("\n \n ENTER TO EXIT BACK TO MENU");
443         Console.ReadKey();
444         ShowMenu();
445     }
446 }
```

Image of code working:



```
Your ships:

------------------
|~|A|B|C|D|E|F|G|H
------------------
|1|~|~|~|~|~|~|~|~
------------------
|2|~|O|~|O|~|~|~|O
------------------
|3|~|~|~|~|~|~|O|~
------------------
|4|~|~|~|~|~|O|O|O
------------------
|5|~|~|~|~|~|O|~|~
------------------
|6|~|O|~|O|O|~|O|~
------------------
|7|~|O|O|O|~|~|O|~
------------------
|8|~|~|~|~|~|O|O|O

Opponent map:

------------------
|~|A|B|C|D|E|F|G|H
------------------
|1|~|~|~|~|~|~|~|~
------------------
|2|~|~|~|~|~|~|~|~
------------------
```

```
------------------
|8|~|~|~|~|~|O|O|O

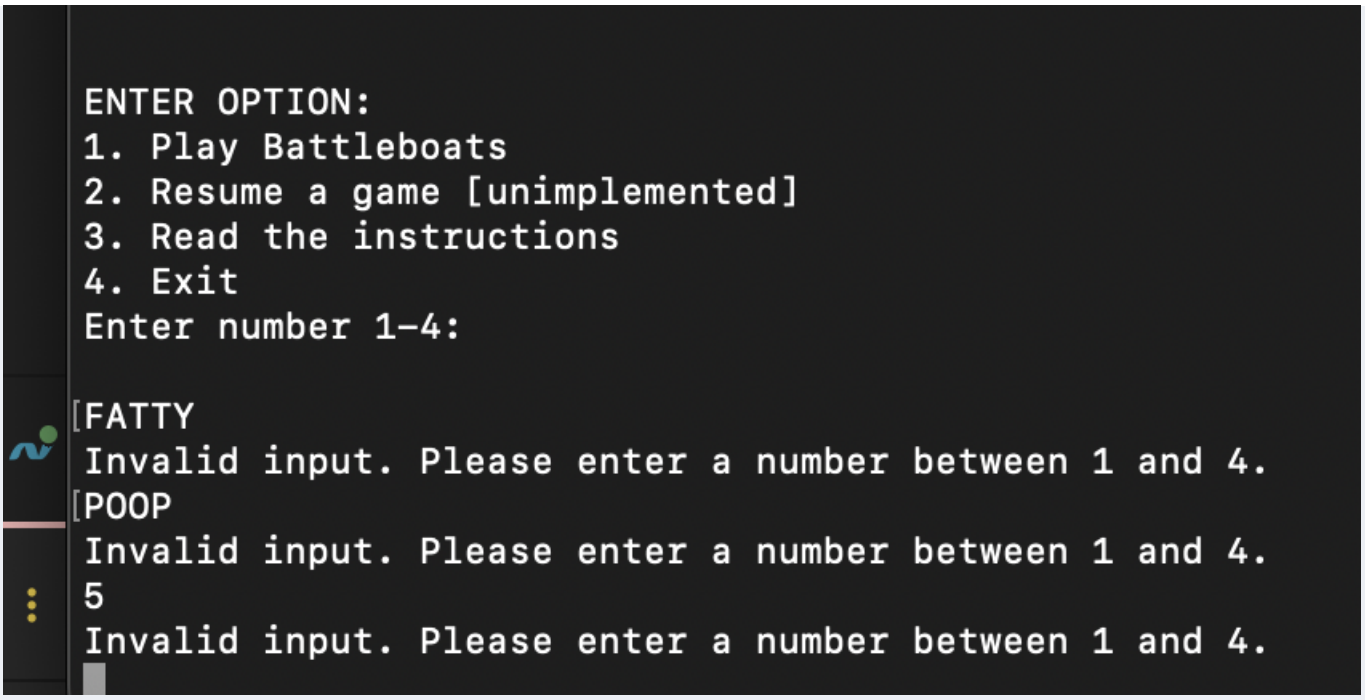
Opponent map:

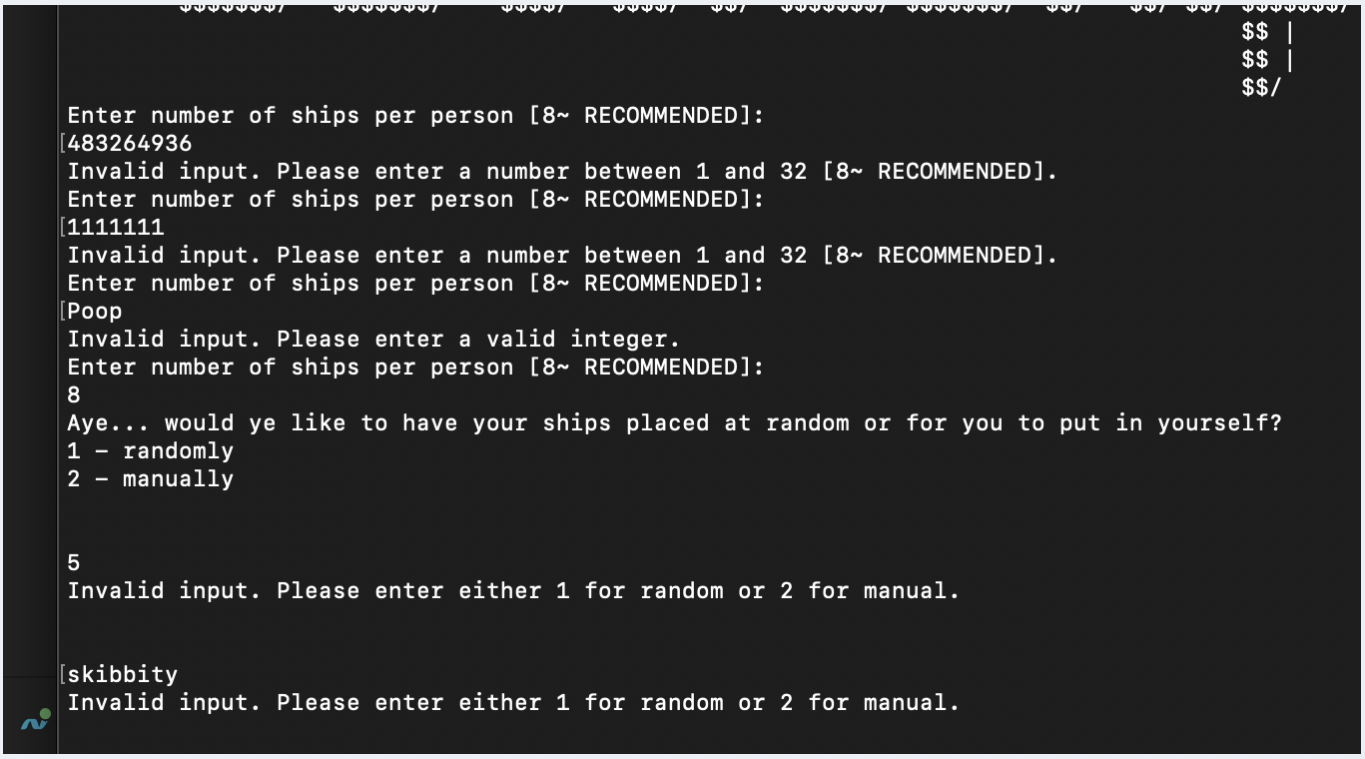
------------------
|~|A|B|C|D|E|F|G|H
------------------
|1|~|~|~|~|~|~|~|~
------------------
|2|~|~|~|~|~|~|~|~
------------------
|3|~|~|~|~|~|~|~|~
------------------
|4|~|~|~|~|~|~|~|~
------------------
|5|~|~|~|~|~|~|~|~
------------------
|6|~|~|~|~|~|~|~|~
------------------
|7|~|~|~|~|~|~|~|~
------------------
|8|~|~|~|~|~|~|~|~

 YOUR TURN CAPTAIN! Enter coords: (ex: B4):


B4
Hit! You sank a ship!
.......The opposing force has managed to sink one of our ships.

Your fleet: 32 | CPU's fleet: 30
```





Evaluation

Success of the Project:

The coding project successfully achieved its criteria. Ship placement, shooting mechanics, and victory conditions were all programmed. The user interface and input validation ensured that no error would slip through the cracks and the end user would have as smooth as an experience as possible when playing, streamlining the game.

Scope Coverage:

The project did everything originally asked minus the challenge tasks which I'm sure I would have completed if given more time. You can see above that I ticked off every single box on the checklist!

Areas for Improvement:

What Went Well:

- Found the coding aspect easy for the most part
- Enjoyed solving coding related problems
- Made the actual game which was good in all its functioning glory

Considerations for Next Time:

- Could make it look nicer (could use colored text for example)
- Add things like multiplayer, player vs player?
- Make it more optimised
- Optimize the code for scalability, in case I want to expand on it :D
- *Hand it in on time*

MILO TEKCHANDANI - 2024