Script programming [Python] (CSCI 6651-02)

**Logo, company name

Description automatically generated**

**PROJECT**

**DEVELOPING OF BATTLESHIP GAME USING PYTHON**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Instructor : Robert Antonetti

Name : Tharangini Venna (00769138)

Dept : Computer Science & information technology

College : University of New Haven

Date : May 6th 2022

**DEVELOPING BATTLESHIP GAME USING PYTHON CODE**

Write a program that simulates the game of Battleship. The game will be completely text-based (see Sample Execution). Battleship is a two-player Navy game. The objective of the game is to sink all ships in your enemy's fleet. The player to sink his/her enemy's fleet first wins. Both players' fleets consist of 5 ships that are hidden from the enemy. Each ship may be differentiated by its "size" (besides the Cruiser , Battleship, Aircraft and Destroyers) or the number of cells it expands on the game board. The Carrier has 5 Ships,2 Destroyers occupying 3 spaces (3 hits to sink),1 Cruiser occupying 5 spaces (5 hits to sink),1 Battleship occupying 7 spaces (7 hits to sink),1 Aircraft Carrier 9 spaces (9 hits to sink)

**Functional Description**

Create a welcome\_screen with word\_battleship that display an initial program welcome to game with battleship word along with the rules of Battleship.

A screenshot of a computer

Description automatically generated

Create a function initialize\_game\_board() that sets each cell in a game board.

A screenshot of a computer

Description automatically generated with medium confidence

* Create a function that allows the user to place each of the 5 types of ships on his/her game board by first player
* Create a function manually\_place\_ships\_on\_board() that places the 5 types of ships on a given board by 2nd player
* Create a function check\_if\_sunk\_ship() that determines if a ship was sunk and function check\_if\_alive\_ship that determine if a ship is alive and also word to print if it hit and word to print if it miss

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Create a function is\_winner() that determines if a winner exists.

A screenshot of a computer

Description automatically generated

Create a function update\_board() that updates the board every time a shot is taken. 'X' indicates a hit and '0' indicates a miss.

A screenshot of a computer

Description automatically generated

Create a file to read and write the high score into a text file

The main function does the following:

o   Opens an output file battleship.txt for writing;

o   Simulates the game of Battleship

o   Outputs data to file

Graphical user interface, text, application, Word

Description automatically generated

The program should be built such that the users are Player1 and the Player2. Two boards exist within the game. Hint: each board should be implemented as 10 rows and 26 columns . One represents Player1's board and one represents Player2's board. At the beginning of the game each player' game board should be initialized to all with their name of rows amd columns as A1,A2,Z1….etc indicating that no ships have been placed on either board. Before the game starts, Player1 should have the option to either manually place each of the 5 ships in his/her fleet or to have them randomly placed on the board. If Player1 decides to place the ships manually, then he/she should be prompted to place the 2 Destroyers first ,1 Cruiser next ,1 Battleship later and then 1 Aircraft Carrier 9 spaces .Note that ships cannot be placed diagonally on the board, they can only be placed vertically or horizontally. You program must check to see if the user tries to place a ship outside the boundaries of the board or on top of a ship that has already been placed. Each cell on the board that contains part of the ship must be indicated by 'c' for Carrier, 'b' for Battleship, 'r' for Cruiser, 's' for Submarine, or 'd' for Destroyer. For example, if the Carrier was placed then the board should contain 5 'c' s for each cell on the board that has a piece of the Carrier, etc. Once Player1's ships have been placed, Player2's ships must be randomly placed. Note that the placement of Player2's ships must be unknown. Thus, Player2's board will only display initial board in each cell after the placement of each ship. The program should randomly select Player1 or Player2 to go first.

Once it has been decided on which player goes first, the game starts. Whenever it's Player1's turn, a prompt should be displayed asking for a position to target (specifying where to "shoot") on the enemy's (Player2's) board . The position should be specified in terms of a row and a column on the board. The row and column should always be displayed along with the board. If the position specified happens to hit a ship, then a 'X' should replace on Player2's board. If the positioned specified misses any one of the ships in the fleet, then a '0' should replace on Player2's board. Note that from turn-to-turn each player should NOT be allowed to enter the same position. Also, between turns clear the screen (system("cls")). In one turn, a player can only take one shot at the enemy's (other player's) fleet. When Player2 takes a shot at Player1's board, each hit should be specified with a 'X' and each miss with a '0' on Player1's board. The game is over win Player1 or Player2 has sunk all of the ships in the fleet of the enemy.

For each move made by Player1 and Player2, the results should be stored to a file called "battleship.txt" file the high score and the low score of each . In this file, you should log the targeted position by each player on each move and whether it was a hit on one of the ships in the fleet. Also, if one of the ships happens to sink, then note this in the log file.

At the end of the game, Player1's and Player2's statistics should be written to "battleship.txt". The stats include the total number of hits, the total number of misses.