Lab #3

CS-2050

February 10, 2024

1 Requirements

In this lab, you will create a 1-D version of the game *Battleship*. The arena will be an array, where each cell (or index) represents a ship. Note that in this version of the game, **each ship is exactly one cell in size**.

You must write the functions required to create and manage the game board, keep track of scoring, and put away (or free) the game board.

1.1 newBoard

```
int * newBoard()
```

Info: This function creates a new empty game board with `BOARD_SIZE` cells, where the user (IE: the main function) can set their own ship positions. The game board must store information hidden from the user, which includes the number of shots taken (as an int), the number of successful hits (as an int), and the score (as a float). The score is calculated by `hits / shots`. The order of the hidden elements is up to you.

An empty cell is represented by a 0 in the array, a ship is represented by a 1 in the array, and a destroyed ship is represented by a -1 in the array.

Value	Meaning
-1	Destroyed shp
0	Empty cell
1	Ship

1.2 takeShot

```
int takeShot(int *board, int cell)
```

Info: This function takes a game board, and a cell (IE: index) to shoot at. If the given cell is occupied by a ship (1), then the cell is changed to a -1 to indicate a hit. If the given cell is either empty (0), or destroyed (-1), then the cell is not changed and no hit occurs.

This function must also update the hidden game values after each shot. It will return 1 on a hit, or 0 on a miss.

1.3 countFreeCells

```
int countFreeCells(int *board)
```

Info: This function takes a board, and returns the number of empty cells on it.

1.4 getShotsTaken

```
int getShotsTaken(int *board)
```

Info: This function takes a board, and returns the number of shots taken (hidden value).

1.5 getHits

```
int getHits(int *board)
```

Info: This function takes a board, and returns the number of successful hits (hidden value).

1.6 getScore

```
float getScore(int *board)
```

Info: This function takes a board, and returns the score (hidden value). **Note that this function must not calculate the score**.

1.7 updateScore

```
void updateScore(int *board)
```

Info: This function takes a board, and updates the score (hidden value) based on the shots and hits.

1.8 endGame

```
void endGame(int *board)
```

Info: This function takes a board, and frees the memory allocated to it.

Grading: 15 points

1. Write required *newBoard* function
   * 5 points
2. Write required *takeShot* function
   * 3 points
3. Write required *countFreeCells* function
   * 2 points
4. Write required *getShotsTaken* function
   * 1 points
5. Write required *getHits* function
   * 1 points
6. Write required *getScore* function
   * 1 points
7. Write required *updateScore* function
   * 1 points
8. Write required *endGame* function
   * 1 points

Notice:

1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the $-Wall$, $-Werror$, and $-Wpedantic$ flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab intro PDF.