CS120 – Computer Science 1
Assignment #8
Spring 2014

In this week's assignment, we recreate the 2-player game "Battleship" as a means of gaining more experience with arrays, and specifically, multi-dimensional arrays. It also serves as an exercise in careful reading and may be done in pairs of two.

This assignment is due on Monday March 31st at 11:59pm.

Rules of the game

Most of you will have played Battleship before at some point, but the rules are summarized here as a bulleted list for your convenience[1]:

- Two players have before them four game boards, two for each player. In this assignment, you only implement the boards for one player. I will call these game boards "board A" and "board B".
- The players cannot see each other's game boards.
- Each board contains a grid of 10 rows by 10 columns.
- At the beginning of the game, each player is given five battleships of different lengths that have to be placed on the player's board A. Once the battleships have been placed, they cannot be moved. The lengths of the battleships are: 2, 3, 3, 4 and 5 positions long. Battleships can be placed horizontally or vertically.
- No position on the board can be occupied by more than one battleship.
- The objective of the game is to guess where the other player has positioned various battleships on his board A before the other player guesses where you placed your battleships on your board A.
- Board B is then used to keep track of which positions you guessed your opponent placed a battleship on.
- In alternating order, each player will name a position by row and column, such as position "two by nine", and the other player will check to see if that position on their board A is occupied by one of their five battleships. If it is, that battleship has been "sunk".
- The previous step repeats until all ships of one player have been sunk.

Implementing Battleship

It is important to see here that each game board can be represented in C++ with a 2-dimensional array: an array with 10 positions (to represent 10 rows), where each

1 See also the wiki on Battleship: http://en.wikipedia.org/wiki/Battleship_(game).

position contains an array with 10 positions (to represent 10 columns). Each player will need at least two such 2-D arrays: one for board A and one for board B.

Board A is used to keep track of where a player's battleships are located. The five ships are placed at random positions on the board. Remember that once a position is occupied by a battleship, other battleships cannot occupy it. Mark the occupied positions on the board with a character that tells you the position is nonempty, such as the character **o** (lowercase o). Mark the unoccupied positions on the board with the character **.** (period).

Hint: use different symbols for different ships, as shown in the example below!

Mark each position on board B with a period.
When the battleships have been placed on board A, a player's boards may look like this:

```
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
```

`Initial board B`

```
. . . . . . . . . .
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . 4 . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .
```

`Initial board A (example)`

The player can now either guess a position on the opponent's game board A, or let the opponent guess a position on their game board.

(1) Guess a position on the opponent's board A

You tell your opponent (verbally) which position you'd like to guess. Enter the position into your own program as X and Y coordinates (or as a row number and column number). Your opponent will then tell you if you have hit one of their battleships or not. The corresponding position on your board B is changed into the character **x** if you've guessed the position of one of your opponent's battleships. Otherwise, change the corresponding position on your board B into the character **o** (lowercase o) to indicate you hit nothing. This way, the player can use board B to estimate where he has the best chance of guessing the location of one of the opponent's battleships in the next turn.

(2) The opponent guesses a position on your board A

Your opponent verbally tells you a position as X and Y coordinates (or a row number and column number) and you enter these coordinates into the program. If the coordinates are occupied by one of your battleships, so if the corresponding position on board A contains a character that is not a period, find a way to keep track of the fact that a ship has been hit that position. Tell your opponent (verbally) if their guess was a hit or a miss.

Hint: there are many ways to store where the opponent has hit your ships in the past. Two examples are: (1) an extra 2-D array of characters that contains only "hit" and "miss" characters, and (2) an array of X and Y coordinates where your ships have been hit in the past.

When the opponent's guess is processed like this, have your program check to see if all your battleships have sunk. If this is true, your program should quit and you should tell your opponent that they have won.

These steps (1) and (2) are done in alternating order by each player. Player 1 performs step (1) while player 2 performs step (2), and when each player is ready the roles are reversed: player 1 performs step (2) and player 2 performs step (1). This goes on until one of the two players admits defeat as part of step (2).

Note: each player has to redraw their game boards after every move!

Beginning of an example program execution

```
-bash-4.1$ ./a.out
Welcome to Battleship, player! Your battleships will now be placed
randomly on your board A.

. . . . . . . . . . . (Board B)
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . 4 . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .

Will you make the first move (1) or your opponent (2)?
1

YOUR TURN:
Please enter coordinates to guess:
2 9
Was this a hit? (y/n)
y

. . . . . . . . . . (Board B)
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. x . . . . . . . .
. . . . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . 4 . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .

OPPONENT'S TURN:
Please enter the coordinates your opponent guesses:
1 9

Board B:
. . . . . . . . . . (Board B)
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. x . . . . . . . .
. . . . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . 4 . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .

YOUR TURN:
Please enter coordinates to guess:
3 10
Was this a hit? (y/n)
n

. . . . . . . . . . (Board B)
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. x . . . . . . . .
. . o . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . 4 . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .

OPPONENT’S TURN:
Please enter the coordinates your opponent guesses:
8 4

. . . . . . . . . . (Board B)
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. x . . . . . . . .
. . o . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . 1 1 1 1 1 . . .
. . . . . . . X . .
. . . . . . . 4 . .
. . . . . . . . . .
. . . . . . . 3 3 3
. 5 5 5 . . . . . .
. . . . . . . . . .
. . . . . 2 2 2 2 .
```

(After several of these turns...)

```
OPPONENT’S TURN:
Please enter the coordinates your opponent guesses:
3 3

. . . . . . . . . . (Board B)
. . . . o x . . . .
. . . . . . . o . .
. . o . . . o . . .
. . . . . . . . . .
. o . . . o . . . .
. . . . . . . x . .
. . . . . . . . . .
. x . . . . . o . .
. . o . . . . . . .

. . . . . . . . . . (Board A)
. . . . . . . . . .
. . x 1 1 1 1 . . .
. . . . . . . x . .
. . . . . . . 2 . .
. . . . . . . . . .
. . . . . . . 3 x x
. 5 5 x . . . . . .
. . . . . . . . . .
. . . . . 2 x 2 2 .

All your battleships have been sunk! You lose the game, but well played
nonetheless. Until next time!
-bash-4.1$
```

Hints and suggestions

- This assignment is as much an exercise about arrays as it is about careful reading. Read through the description step by step, write down which variables you think you will need and look at the example execution to see what kind of interaction with the user you should expect to include.

- A lot of small details are left open for you to solve, such as how you check if all your battleships have been sunk. You will need creativity to solve these problems.
- Keep your output as concise as possible to make your program easier to debug. If you do the assignment with a partner, make the program's output look as much like the example output as you can to make collaboration with your partner easier.
- This assignment does not have to be done in pairs, but doing so will make debugging your application easier. Two pairs of eyes see errors more easily than one, especially when you and your partner are testing both your implementations at the same time over a game of Battleship.
- Feel free to ask your TA for help if any part of the assignment is unclear or if you're unsure how to solve a problem, but also remember to make use of the experience of your classmates through the CS forums! The forums can be found here: https://forums.cs.uidaho.edu. You will be asked to log in with the same credentials you would use to log into the Wormulon server.

Submitting your solution

When your program works as expected, turn it in online through *cscheckin* after concatenating all source code files (.cpp), header files (.h) and a typescript file. Additionally, submit a hardcopy of your source code files and header files by running it through an online *pretty printer*: go to http://hilite.me, paste your source code in the "Source code" box (one source code file at a time), select the right language (C++), and click the "Highlight!" button. Finally, print the text under the word "Preview".

N.B.: the cscheckin command is **cscheckin –f filename –c coursename**, where "filename" is the name of the file you want to submit and "coursename" is "cs120".