Programming Paradigms 2013-2014
Haskell Assignment
Battleship

Naomi Christis
Office G028
naomi.christis@uantwerpen.be

Your assignment is to build and test a haskell implementation of the classic game battleship.

1 Battleship

Your program should fulfill the following requirements:

- The playing field should measure 10 by 10 cells.
- A game has two players.
- Each player's fleet has four ships: one with length 2, one with length 3, one with length 4 and one with length 5. A ship with length l takes up l cells on the grid, the cells are arranged either horizontally next to each other, or vertically above each other.
- Each player's goal is to sink his opponent's battleships. A player's ship is sunk when his opponent has called out each of the cells belonging to that ship. The game ends when one player's entire fleet is sunk.
- Ships belonging to the same player are not allowed to share cells.
- Players take turns calling out coordinates on the other player's field. Each turn, a player can fire as many shots as he has ships that aren't sunk.
- Each turn, the program should print out which coordinates the player calls out. It should also announce whether or not the coordinate that was called out was occupied by one of the opponent's ships. When a player's ship is sunk, the program should print "You sunk my battleship!". When the game has ended, the winner should be announced and both playing fields should be displayed, including which cells were targeted.
- Exception handling for the most obvious errors i.e. overlapping ships, wrong input (too many ships, ship that do not lie on a row, ...), ...

as well as the input format defined in the next section.

Reminder: The goal of this assignment isn't to bludgeon a procedural or object oriented solution into Haskell, but rather to write idiomatic Haskell code that elegantly solves the problem.

2 Input format

You must adhere *exactly* to the given input format. Your program should read from standard input, not from a file on your hard drive.
On the left follows an example of the input needed to play a partial game of battleship. The column on the right explains what each line on the left does.

`An`	First player's name
`Bert`	Second player's name
`(0,0);(0,1)`	Position of the cells of An's ship with length 2
`(6,7);(7,7);(8,7)`	Position of the cells of An's ship with length 3
`(3,3);(3,4);(3,5);(3,6)`	Position of the cells of An's ship with length 4
`(5,9);(6,9);(7,9);(8,9);(9,9)`	Position of the cells of An's ship with length 5
`(6,6);(6,7)`	Position of the cells of Bert's ship with length 2
`(3,2);(3,3);(3,4)`	Position of the cells of Bert's ship with length 3
`(9,6);(9,7);(9,8);(9,9)`	Position of the cells of Bert's ship with length 4
`(4,9);(5,9);(6,9);(7,9);(8,9)`	Position of the cells of Bert's ship with length 5
`(5,5)`	An fires the first shot at coordinates (5,5)
`(6,7)`	An fires the second shot at coordinates (6,7) (Hit)
`(5,6)`	An fires the third shot at coordinates (5,6)
`(1,1)`	An fires the fourth shot at coordinates (1,1)
`(6,7)`	Ben fires the first shot at coordinates (6,7) (Hit)
`...`	...

As can be seen in the previous example, different coordinates are separated with semi-colons. Coordinates are expressed in the form (horizontal, vertical) and each coordinate is zero-indexed.

CAUTION:You should accept the input in the left column *exactly*.

3 Practical

Submit your solution before the imposed deadline through Blackboard in a zip archive. No solutions will be accepted via e-mail; only timely submissions posted on BlackBoard will be accepted and assessed; no extensions of the deadline will be granted. You are expected to work on this assignment individually. Recall that work submitted for grading must ultimately be your own work, reflecting your personal learning curve and performance. Cheating is a serious academic offense; we do not tolerate cheating, nor assisting others to do so.