The Game - Battleship

The game is played on a ***4×8 grid***, and involves one player, the ***searcher*** trying to find the locations of three battleships hidden by the other player, the ***hider***.

After each ***guess***, the hider responds with three numbers:

- the number of ships ***exactly*** located
- the number of guesses that were exactly ***one*** space away from a ship
- the number of guesses that were exactly ***two*** spaces away from a ship

Each guess is only counted as its closest distance to any ship.

The eight squares adjacent to a square, including diagonally adjacent, are counted as distance 1 away. The sixteen squares adjacent to those squares are considered to be distance 2 away. Some of these locations will be outside the board.

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

This feedback does not tell you *which* of the guessed locations is close to a ship.

Chess board

	A	B	C	D	E	F	G	H
1	A1	B1	C1	D1	E1	F1	G1	H1
2	A2	B2	C2	D2	E2	F2	G2	H2
3	A3	B3	C3	D3	E3	F3	G3	H3
4	A4	B4	C4	D4	E4	F4	G4	H4

A few caveats:

- The three ships will be at three ***different*** locations.
- Your guess must consist of exactly three ***different*** locations.
- Your list of locations may be written in any order, but **the order is not significant**; the guess *A3, D1, H1* is exactly the same as *H1, A3, D1* or any other permutation.

Examples:

Locations	Guess	Feedback
H1, B2, D3	*B3, C3, H3*	0, 2, 1
H1, B2, D3	*B1, A2, H3*	0, 2, 1
H1, B2, D3	*B2, H2, H1*	2, 1, 0
A1, D2, B3	*A3, D2, H1*	1, 1, 0
A1, D2, B3	*H4, G3, H2*	0, 0, 0
A1, D2, B3	*D2, B3, A1*	3, 0, 0

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
1								**S**
2		**S**						
3		**G**	**G**	**S**				**G**
4								

The game finishes once the searcher guesses all three ship locations in a single guess (in any order), such as in the last example above. The object of the game for the searcher is to **find the target with the fewest possible guesses.**

The Program

Function:

- return your initial guess (called once)
- use the feedback from the previous guesses to determine the next guess
- determine the feedback to give to hider, given his target and a target

You will find it useful to keep information between guesses; since Haskell is a purely functional language, you cannot use a global or static variable to store this. Therefore, **your initial guess function must return this game state information**, and **your next guess function must take the game state as input and return the new game state as output.**

You may put any information in the game state, but you must define a ***type GameState*** to hold this information. If you do not need to maintain any game state, you may simply define type GameState = ().

You must also define a ***type Location*** to represent grid locations in the game, and you must **represent your guesses as lists of Locations.** Your Location type must be an instance of the Eq type class. Of course, two Locations must be considered equal if and only if they are identical. You must also define a function to convert a Location into a two-character string of the upper-case column letter and row numeral, as shown throughout this document.

What you must define

In summary, in addition to defining the `GameState` and `Location` types, you must define following functions:

toLocation :: String → Maybe Location
: gives `Just` the `Location` named by the string, or `Nothing` if the string is not a valid location name.

fromLocation :: Location → String
: gives back the two-character string version of the specified location; for any location *loc*, `toLocation (fromLocation` *loc*`)` should return `Just` *loc*.

feedback :: [Location] → [Location] → (Int,Int,Int)
: takes a target and a guess, respectively, and returns the appropriate feedback, as specified above.

initialGuess :: ([Location],GameState)
: takes no input arguments, and returns a pair of an initial guess and a game state.

nextGuess :: ([Location],GameState) → (Int,Int,Int) → ([Location],GameState)
: takes as input a pair of the previous guess and game state, and the feedback to this guess as a triple of the number of correct locations, the number of guesses exactly one square away from a ship, and the number exactly two squares away, and returns a pair of the next guess and new game state.