universität innsbruck

Functional Programming | WS 2019/2020 | LVA 703024+703025

Exercises 12 | Deadline: January 22, 2020, 6am

- Please write all your Haskell functions from this exercise sheet into a single .hs-file and upload it in OLAT.
- You can use a template .hs-file that is provided on the proseminar page.
- The file should compile with ghci.
- Once the file has been uploaded, it cannot be changed or resubmitted!

Exercise 12.1 *Connect Four* **10 p.**

In this exercise we want to extend the implementation of Connect Four from the lecture in various ways. Note that all sub-tasks can be solved independently.

1. Modify the function `winning_player` in the game logic, so that also diagonals are taken into account. (2 points)

2. The user-interface does not check whether input moves are valid: it is not checked whether the input from the user really is a number, and whether this number is a valid move. Both cases may lead to unintended behavior or crashes of the programs. Therefore, you should modify the user-interface in a way that it repeatedly asks for input until a valid move has been entered, e.g. as follows:

   ```
   Choose one of [0,1,2,3,4,5,6]: five
   five is not a valid move, try again: 8
   8 is not a valid move, try again: 3
   ... accept and continue ...
   ```

   (2 points)

3. Modify the user interface so that after a match has been completed, it asks whether another round should be played. If so, the starting player should be switched. Clearly, this also requires a change in the type of `initial_state`. (2 points)

4. Extend the implementation so that it can save and load games, e.g., via file `connect4.txt`. The user interface might look like this:

   ```
   Welcome to Connect Four
   (n)ew game or (l)oad game: l
   ... game starts by loading state from connect4.txt ...
   Choose one of [0,2,3,5,6] or (s)ave: s
   ... game is saved in file connect4.txt and program quits ...
   ```

   For the implementation, note that `read . show = id` and that one can automatically derive `Read`-instances in datatype definitions. (2 points)

5. Extend the implementation so that it can give hints. To be more precise, the user interface should inform the current player, whenever she can win within 1 or 2 moves by providing a hint. Winning in 2 moves means that after following the move from the hint, you will win the game no matter how the opponent moves in between your moves. In that case the player can type ”`h`” to see a first move that leads to success.

```
Choose one of [0,2,3,5,6] or see (h)int to win within 2 moves: h
Hint: Drop a piece in column 2
Choose one of [0,2,3,5,6]: 3
... the game continues since the user is not forced to follow hints
```

(2 points)