universität innsbruck

Funktionale Programmierung	WS 2025/2026	LVA 703025
Übungsblatt 10, 10 Punkte		Abgabefrist: Mittwoch, 7. Januar 2026, 6 Uhr

- Kreuzen Sie Ihre bearbeiteten Aufgaben im OLAT-Kurs des PS an.
- Sie können von der Connect-Four-Implementierung aus der Vorlesung ausgehen, d. h. `Main.hs` und `Logic.hs` aus diesem Archiv: http://cl-informatik.uibk.ac.at/teaching/ws25/fp/slides/Scripts_10.zip.
- Laden Sie Ihre modifizierten Dateien `Main.hs` und `Logic.hs` in OLAT hoch. Zippen Sie die Dateien nicht, sondern laden Sie jede Datei einzeln hoch.
- Ihre .hs-Dateien müssen mit ghci kompilierbar sein.

Aufgabe 1 *Vier gewinnt* **10 P.**

In dieser Übung wollen wir die Implementierung von Connect Four aus der Vorlesung auf verschiedene Arten erweitern. Beachten Sie, dass alle Teilaufgaben unabhängig voneinander gelöst werden können.

1. Die Benutzeroberfläche prüft nicht, ob eingegebene Züge gültig sind: Es wird nicht geprüft, ob die Eingabe des Benutzers wirklich eine Zahl ist und ob diese Zahl ein gültiger Zug ist. Beide Fälle können zu unbeabsichtigtem Verhalten oder Abstürzen des Programms führen. Daher sollten Sie die Benutzeroberfläche so anpassen, dass sie wiederholt nach einer Eingabe fragt, bis ein gültiger Zug eingegeben wurde, z. B. wie folgt:

   ```
   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 Punkte)

2. Modifizieren Sie die Benutzeroberfläche so, dass nach Abschluss einer Partie gefragt wird, ob eine weitere Runde gespielt werden soll. Wenn ja, sollte der startende Spieler gewechselt werden. Dies erfordert offensichtlich auch eine Änderung im Typ von `initState`. (2 Punkte)

3. Erweitern Sie die Implementierung so, dass sie Spiele speichern und laden kann, z. B. über die Datei `connect4.txt`. Die Benutzeroberfläche könnte so aussehen:

   ```
   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 ...
   ```

   Beachten Sie für die Implementierung, dass `read . show = id` gilt und dass man `Read`-Instanzen in Datentyp-Definitionen automatisch ableiten kann. (2 Punkte)

4. Modifizieren Sie die Funktion `winningPlayer` in der Spiellogik, sodass auch Diagonalen berücksichtigt werden.

   (2 Punkte)

5. Erweitern Sie die Implementierung durch Hinzufügen von Hinweisen. Genauer gesagt sollte die Benutzeroberfläche den aktuellen Spieler informieren, wann immer er oder sie innerhalb von 1 oder 2 Zügen gewinnen kann, indem ein Hinweis bereitgestellt wird. Gewinnen in 2 Zügen bedeutet, dass man nach Befolgen des Zuges aus dem Hinweis das Spiel gewinnen wird, egal wie der Gegner zwischen den Zügen zieht. In diesem Fall kann der Spieler "`h`" eingeben, um einen ersten Zug zu sehen, der zum Erfolg führt.

```
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 Punkte)