Amőba

Feladat ismertetése

A feladatunk egy teljes amőba játékot készíteni, amellyel két játékos játszhat n×n-es méretű táblán.

Az amőba játékszabályai a következők:

Az amőba ezen változatához egy tetszőleges méretű négyzet alakú tábla szükséges. A két játékos két egymástól eltérő szimbólumot használ a saját mezőjük elfoglalására (ezek klasszikusan `X` és `O` jelek). A játékot az a játékos kezdi, aki az `X` szimbólumot használja. A két játékos egymás után felváltva helyezi a saját szimbólumát a tábla egy még üres mezőjére. A játék célja, hogy a táblán egy teljes sor, teljes oszlop, a főátló vagy a mellékátló közül valamelyik játékos szimbólumával legyen tele. Ha ez megtörténik, akkor az a játékos nyer, akinek a szimbóluma szerepel a felsoroltak valamelyikében.

Főátló

Egy tábla főátlóját az alábbi példa szemlélteti:

$$\begin{bmatrix} X & \_ & \_ \\ \_ & X & \_ \\ \_ & \_ & X \end{bmatrix}$$

Mellékátló

Egy tábla mellékátlóját az alábbi példa szemlélteti:

$$\begin{bmatrix} \_ & \_ & O \\ \_ & O & \_ \\ O & \_ & \_ \end{bmatrix}$$

Szükséges típusok

Az alábbi saját típusokat másoljuk be a `TicTacToe.hs` fájlba.

```haskell
data Player = X | O deriving Eq
data Cell = E | P Player deriving Eq
data State = Running | GameOver (Maybe Player) deriving Eq

type Size = Int
type Board = [[Cell]]
type Game = (Size, Board, Player, State)
```

Rövid magyarázat:

- `Player`: A két játékost írja le, melyek értéke `X` és `O` lehet.
- `Cell`: A mező státuszát jelöli, amely lehet:

  - `E`: Mező még nem foglalt, üres.
  - `P X`: A mezőn az `X` játékos jele szerepel.
  - `P O`: A mezőn a `O` játékos jele szerepel.

- `State`: Ezzel adjuk meg, hogy a játék folyik-e még vagy már vége van és mi lett az eredménye.
  - `Running`: A játék még fut.
  - `GameOver (Maybe Player)`: A játéknak vége, `Nothing` esetén döntetlen, ellenkező esetben a megadott játékos győzött.

- `Board`: A táblát listák listájával ábrázoljuk. A listák a tábla sorait jelölik, a sorok pedig `Cell`-ekből állnak.

- `Game`: Típusszinoníma, ez adja meg a egész játékállást, valamennyi lépésben ezt az állapotot kell változtatni. Tartalmazza a tábla méretét, magát a táblát, a következő játékost és a jelenlegi játékállapotot.

Első szint: Elem kicserélése egy táblán (2 pont)

Adjuk meg azt a függvényt, amely az adott indexen lévő elemet lecseréli a paraméterben megadottra egy listák listájában. A rendezett pár első komponense a tábla sorát, míg a második a soron belüli elemet jelöli. Ha az indexelés helytelen, akkor a lista ne változzon!

```
replaceAtMatrix :: (Int,Int) -> a -> [[a]] -> [[a]]

replaceAtMatrix (-1,0) 'A' ["bcd","lds","rtz"] == ["bcd","lds","rtz"] -- -1 túl kicsi
replaceAtMatrix (0,0) 'A' ["bcd","lds","rtz"] == ["Acd","lds","rtz"]
replaceAtMatrix (2,0) 'A' ["bcd","lds","rtz"] == ["bcd","lds","Atz"]
replaceAtMatrix (3,0) 'A' ["bcd","lds","rtz"] == ["bcd","lds","rtz"] -- 3 túl nagy
replaceAtMatrix (2,2) 'A' ["bcd","lds","rtz"] == ["bcd","lds","rtA"]
replaceAtMatrix (2,3) 'A' ["bcd","lds","rtz"] == ["bcd","lds","rtz"] -- 3 túl nagy
replaceAtMatrix (2,-10) 'A' ["bcd","lds","rtz"] == ["bcd","lds","rtz"] -- -10 túl kicsi
take 5 (replaceAtMatrix (1,2) 100 [[1..],[1,11..],[],[12,13,14]] !! 1) == [1,11,100,31,41]
replaceAtMatrix (3,1) X [[O,O,O],[O],[],[O,O,O]] == [[O,O,O],[O],[],[O,X,O]]
```

Második szint: Győztes tábla alapján (5 pont)

Mondjuk meg, hogy ki lehet a győztes egy adott állapotban. A függvénynek ellenőriznie kell, hogy a tábla valamelyik sorának, oszlopának, főátlójának vagy mellékátlójának valamelyikén ugyanazon játékos jelei vannak-e. Ha igen, adjuk vissza azt a játékost, akinek a jelei szerepelnek az adott helyen. Ha nincs ilyen, úgy adjunk vissza `Nothing`-ot.

```
winner :: Game -> Maybe Player
```

```
winner (3,[[P X, P X, P O],[P X,P O,P O],[P X, E, E]],X,Running) == Just X
winner (3,[[P X, E, P O],[E,P X,P O],[P X, E, E]],O,Running) == Nothing
winner (3,[[P X, P X, E],[P O,P O,P O],[P X, E, E]],O,Running) == Just O
winner (3,[[P X, P O, P O],[P O,P X,P O],[P X, E, P X]],O,Running)  == Just X
winner (3,[[P X, P O, P O],[P X,P O,P X],[P O, E, P X]],X,Running) == Just O
winner (4,[[P X,P O,E,P X],[P O,P O,P X,P X],[P O, P X, E, E],[P X, P O, E, E]],
        O,Running) == Just X
```

Harmadik szint: Egy előre megadott játék lejátszása (10 pont)

Adjuk meg azt a függvényt, amely egy előre megadott lépéssorozatot végrehajt a kezdő állásból indulva egy adott méretű táblán. A játék (a `Game` típusú érték) kezdetben az alábbiakból áll:

- A tábla mérete. Ezt paraméterül kapja a függvény és ez az érték nem változik a játék során. Ettől az értéktől függ a tábla mérete.
- Maga a tábla (`Board` típusú érték), amelynek kezdetben minden mezője üres. Például ha a tábla mérete 4, akkor a táblának 4 sora lesz és minden sorban 4 üres mező található.
- A soron következő játékos, ez kezdetben az `X` játékos.
- A játék állapota, amely kezdetben `Running`.

A végrehajtás közben minden lépésből adódó táblaállapotot tartalmaznia kell az eredménynek. Ha a játéknak vége van, tehát van nyertes vagy tele van a tábla és nincs nyertes, akkor a játék kerüljön `GameOver` állapotba, paraméterként kapja meg az eredményt (`Nothing`, ha döntetlen; `Just X`, ha `X` nyert; `Just O`, ha `O` nyert). A játék a `GameOver` állapotában a további lépések hatására a játék állása, nyertese, állapota semmilyen módon nem változhat.

```
gameProgress :: Size -> [(Int,Int)] -> [Game]

gameProgress 3 [] == [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running)]
gameProgress 3 [(1,1)] == [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                           (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running)]
gameProgress 3 [(1,1),(2,2)] == [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                                 (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running),
                                 (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running)]
gameProgress 3 [(1,1),(2,2),(2,3)] ==
               [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running)]
gameProgress 3 [(1,1),(2,2),(2,3),(1,1)] ==
               [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running),
```

```
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running)]
gameProgress 3 [(1,1),(2,2),(2,3),(1,1),(0,0)] ==
               [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[P X,E,E],[E,P X,E],[E,E,P O]],O,Running)]
gameProgress 3 [(1,1),(2,2),(2,3),(1,1),(0,0),(2,1),(2,0),(1,0),(0,2),(0,1),(1,2)] ==
               [(3,[[E,E,E],[E,E,E],[E,E,E]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,E]],O,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[E,E,E],[E,P X,E],[E,E,P O]],X,Running),
                (3,[[P X,E,E],[E,P X,E],[E,E,P O]],O,Running),
                (3,[[P X,E,E],[E,P X,E],[E,P O,P O]],X,Running),
                (3,[[P X,E,E],[E,P X,E],[P X,P O,P O]],O,Running),
                (3,[[P X,E,E],[P O,P X,E],[P X,P O,P O]],X,Running),
                (3,[[P X,E,P X],[P O,P X,E],[P X,P O,P O]],O,GameOver (Just X)),
                (3,[[P X,E,P X],[P O,P X,E],[P X,P O,P O]],O,GameOver (Just X)),
                (3,[[P X,E,P X],[P O,P X,E],[P X,P O,P O]],O,GameOver (Just X))]
gameProgress 4 [(1,1),(2,2)] ==
               [(4,[[E,E,E,E],[E,E,E,E],[E,E,E,E],[E,E,E,E]],X,Running),
                (4,[[E,E,E,E],[E,P X,E,E],[E,E,E,E],[E,E,E,E]],O,Running),
                (4,[[E,E,E,E],[E,P X,E,E],[E,E,P O,E],[E,E,E,E]],X,Running)]
```

Bónusz feladat

A következő feladat nem kerül pontozásra!

A játék konzolból játszható, amelyhez a kód a mellékelt `Main.hs` fájlban található! Illetve a lentebb található kiegészítéseket meg kell tenni.

Modul létrehozása

Modult az alábbi minta alapján tudunk létrehozni:

```
module ... where

import ...
import ...
```

A `module` és a `where` kulcsszavak között megadhatjuk a saját modulunk nevét. Fontos, hogy a mintában szereplő kódrészletnek a fájl legtetején kell szerepelnie,

csak ezt követően jöhetnek az importok, daták, függvények és minden más nyelvi elem. A modul nevét nagy kezdőbetűvel írjuk és fontos, hogy konvenció szerint meg kell egyeznie a kiterjesztés nélküli fájl nevével.

Ezen ismeretek birtokában hozzunk létre egy modult `TicTacToe` néven!

Lépések és játék vége

Ahhoz, hogy valójában játszható legyen a konzolból a játék, szükséges megadni három függvényt: `playerTurn :: Game -> (Int, Int) -> Game`, `isGameOver :: Game -> Bool`, `initGame :: Size -> Game`. A `playerTurn` függvény egy játékos lépését hajtja végre a játékban, míg az `isGameOver` függvény ellenőrzi, hogy a játék `GameOver` állapotban van-e, az `initGame` függvény egy adott méretű táblához állítja elő a kezdőállapotot a "harmadik szintű" feladatban leírtaknak megfelelően.

Main.hs betöltése

Ha kész a játék, akkor a `Main.hs` fájlt kell betölteni. Mind a `TicTacToe.hs` és a `Main.hs` fájlnak azonos mappán belül kell lennie. Ha az megvan, akkor a ghci-t az azonos mappából indítva a `:l Main.hs` fájlt betöltve fordulnia kell és a játék a `main` függvény indításával játszható lesz. Ha nem azonos mappából töljük be a ghci-t, akkor mindkét fájlt szükséges betölteni, mindkét fájlt meg kell adni a `:l`-nek, szóközzel elválasztva egymástól (ez lehet akár relatív, akár abszolút útvonal is), például Windows-on: `:l Feladat\TicTacToe.hs Feladat\Main.hs`, Unix-on: `:l /home/User/Feladat/TicTacToe.hs /home/User/Feladat/Main.hs`. Ekkor lehet, hogy még importálni kell a nézetet a ghci-be, ha véletlenül ez áll fenn, akkor tegyük meg a ghci-ben: `import Main`. Ez után a `main` függvényt meghívva játszható a játék.

A program egy játékostól vár két indexet szóközzel elválasztva. Először a sornak az indexét, majd az oszlopét 0-tól kezdve az indexelést.