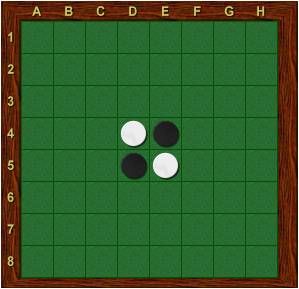
Reversi  
foldl (\acc x -> if x == 1 then True acc) False [1,1,1,1,0,0,0,3,3,3]

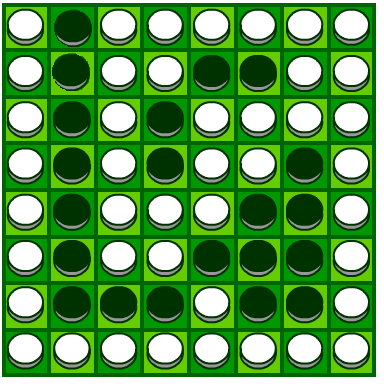
Reversi is a strategy board game for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces called disks, which are white on one side and black on the other. Players take turns placing discs on the board with their assigned color facing up.

During a play, any discs of the opponent's color that are in a straight line and bounded by the disc just placed and another disc of the current player's color are turned over to the current player's color.

Here is an example of an initial board:



And here is an example of an end:



A. Define Datatypes

**data** Reversi = Reversi {rows :: [[Maybe Char]]}

For example:

example:: Reversi

example =

Reversi

[[Nothing, Nothing, Just W, Just W, Just W, Nothing, Nothing, Nothing]

,[Nothing, Nothing, Nothing, Just W, Nothing, Just W, Nothing, Just W]

,[Nothing, Nothing, Just W, Just B, Just B, Just B, Nothing, Just W]

,[Nothing, Nothing, Nothing, Just B, Just B, Just B, Just B, Just W]

,[Nothing, Nothing, Nothing, Just B, Just B, Just B, Nothing, Nothing]

,[Nothing, Nothing, Nothing, Nothing, Just B, Just B, Just B, Nothing]

,[Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing]

,[Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing]

]

This is an example of a Reversi board, in which Just B represents the Black, Just W represents the White.

We need another datatype to represent the position of a disk.

**data** DisksPosition = (Char, Int)

B. Printing Reversi

The following is an example text-representation that we will use. It actually represents the example above. O represents the White, while W represents the Black.

12345678

A..OOO...

B...O.O.O

C..OXXX.O

D...XXXXO

E...XXX..

F....XXX.

G........

There are 8 lines of text in this representation, each corresponding to a row. Each line contains 8 characters. A digit 1 – 8 and A - G represents a the position of a disk.

|  |
| --- |
| printReversi :: Reversi -> IO ()  that, given a Reversi, creates instructions to print the Reversi on the screen, using the format shown above.  Example:  **Reversi>** printReversi initialBoard  12345678  A........  B........  C........  D...XO...  E...OX...  F........  G........ |

C.Some Basic Functions

**C1.**Implement a function

initialBoard :: Reversi

that represents an initial board.

**C2.**Give a position and one kind of disk, check if the position is available for the disk.

isReversi :: DisksPosition -> Maybe DiskColor -> Reversi -> Bool

**C3.**When playing the game, we input the position of the disk, which will return a new Reversi.

inputDisk :: DisksPosition -> Maybe Char -> Reversi -> Reversi

Example:

**Reversi>** inputDisk (C,4) (Just w) initialBoard

12345678

A........

B........

C...O....

D...XO...

E...OX...

F........

G........

**C4.**When a new disk is input, check if there is any turning over should be done, then returns a new Reversi.

Turnover :: Reversi -> Reversi

Example:

**Reversi>** turnOver $ inputDisk (C,4) (Just w) initialBoard

12345678

A........

B........

C...O....

D...OO...

E...OX...

F........

G........

**C5.**Find all available blanks.

blanks :: DiskColor -> Reversi -> [DisksPosition]

D. Game Over and Winner

**D1.** Give a Reversi, is the game finished?

gameOver :: Reversi -> Bool

**D2.** To tell who is the winner

winner :: Reversi -> Player

Here Player is a new data type:

data Player = BlackPlayer | WhitePlayer

deriving (Show,Eq)

E. Game Loop

Write a file to control the game loop.