1 2048

1.1 Aufgabe

Schreibe eine Datenklasse `GameState`. Diese hat die folgenden Eigenschaften:

- `board` - das Spielfeld
- `score` - die Punktzahl des Spielers
- `gameWon` - gibt an, ob der Spieler das Spiel bereits gewonnen hat.

```
GameState(listOf(listOf(2, 2),
                 listOf(0, 4)),
          10, false)
```

```
GameState(board=[[2, 2], [0, 4]], score=10, gameWon=false)
```

1.2 Aufgabe

Implementiere eine Funktion `showGameState`. Diese gibt eine Darstellung des Spielfelds und der aktuellen Punktezahl als Liste von Strings zurück.

```
showGameState(GameState(listOf(listOf(0, 2, 4),
                               listOf(2, 2, 2),
                               listOf(2, 4, 2)),
                        4, false))
```

```
[|    0 |    2 |    4 |, |    2 |    2 |    2 |, |    2 |    4 |    2 |, Score: 4]
```

Hinweis: Nutze `showRows`!

1.3 Aufgabe

Implementiere eine Datenklasse `RowAndScoreIncrease`. Diese hat eine Eigenschaft `row`, die als Liste von Integern dargestellt wird. Die zweite Eigenschaft hat den Namen `scoreIncrease` und den Typ Integer.

```
RowAndScoreIncrease(listOf(2, 4), 0)
```

```
RowAndScoreIncrease(row=[2, 4], scoreIncrease=0)
```

1.4 Aufgabe

Ändere die Funktion `moveRowLeftHelper`. Dieser wird eine Liste mit Integern übergeben. Du kannst davon ausgehen, dass diese nur 2-er Potenzen, die mindestens 2 sind, enthält.

Die Funktion gibt immer noch die Liste der 2-er Potenzen zurück, die beim Kippen nach Links im Spiel 2048 entsteht. Sie gibt auch zurück, wie viele Punkte der Spieler durch die Bewegung dieser Zeile bekommt.

Der Spieler bekommt Punkte, wenn zwei Blöcke kombiniert werden. Seine Punktzahl erhöht sich um die Summe der kombinierten Blöcke.

```
1 moveRowLeftHelper(listOf(2))
```

```
RowAndScoreIncrease(row=[2], scoreIncrease=0)
```

```
1 moveRowLeftHelper(listOf(2, 2))
```

```
RowAndScoreIncrease(row=[4], scoreIncrease=4)
```

```
1 moveRowLeftHelper(listOf(2, 2, 4))
```

```
RowAndScoreIncrease(row=[4, 4], scoreIncrease=4)
```

```
1 moveRowLeftHelper(listOf(2, 2, 2))
```

```
RowAndScoreIncrease(row=[4, 2], scoreIncrease=4)
```

1.5 Aufgabe

Ändere die Funktion `moveRowLeft`. Dieser wird eine Liste mit Integern übergeben.

Die Funktion gibt immer noch die Zeile zurück, die beim Kippen nach Links im Spiel 2048 entsteht. Diese Liste die zurückgegeben wird, ist genauso lang, wie die Liste die übergeben wird. Sie gibt auch zurück, wie viele Punkte der Spieler durch die Bewegung dieser Zeile bekommt.

```
1 moveRowLeft(listOf(0, 0, 0, 2))
```

```
RowAndScoreIncrease(row=[2, 0, 0, 0], scoreIncrease=0)
```

```
1 moveRowLeft(listOf(0, 2, 0, 2, 0))
```

```
RowAndScoreIncrease(row=[4, 0, 0, 0, 0], scoreIncrease=4)
```

```
moveRowLeft(listOf(0, 2, 2, 4, 0))
```

```
RowAndScoreIncrease(row=[4, 4, 0, 0, 0], scoreIncrease=4)
```

```
moveRowLeft(listOf(0, 2, 2, 2))
```

```
RowAndScoreIncrease(row=[4, 2, 0, 0], scoreIncrease=4)
```

```
moveRowLeft(listOf(4, 4, 4, 4))
```

```
RowAndScoreIncrease(row=[8, 8, 0, 0], scoreIncrease=16)
```

Hinweis: Nutze `removeZeros`, `moveRowLeftHelper` und `fill`!

1.6 Aufgabe

Ändere die Funktion `moveBoardLeft`. Dieser wird der aktuelle Zustand des Spiels übergeben.

Die Funktion gibt neuen Zustand des Spiels, nach dem Kippen nach links zurück. Dabei kann sich die Punktzahl ändern und das Spiel gewonnen werden.

```
moveBoardLeft(GameState(listOf(listOf(2, 2),
                               listOf(0, 4)),
                        10, false))
```

```
GameState(board=[[4, 0], [4, 0]], score=14, gameWon=false)
```

```
moveBoardLeft(GameState(listOf(listOf(2, 2, 4),
                               listOf(1024, 1024, 2),
                               listOf(2, 4, 2)),
                        4, false))
```

```
GameState(board=[[4, 4, 0], [2048, 2, 0], [2, 4, 2]], score=2056, gameWon=true)
```

Hinweis: Nutze `moveRowLeft` und `contains2D`!

1.7 Aufgabe

Ändere die zweite Funktion mit dem Namen `rotateRight`. Der Funktion wird der Zustand des Spiels und ein Integer n übergeben. Sie gibt den neuen Zustand des Spiels zurück, nachdem das Spielfeld n-mal nach rechts gedreht wurde. Die beiden anderen Werte ändern sich nicht.

```
rotateRight(GameState(listOf(listOf(1, 2, 3),
                             listOf(4, 5, 6),
                             listOf(7, 8, 9)),
                      10, false), 2)
```

```
GameState(board=[[9, 8, 7], [6, 5, 4], [3, 2, 1]], score=10, gameWon=false)
```

```
rotateRight(GameState(listOf(listOf(2, 2),
                             listOf(0, 4)),
                       8, true) , 3)
```

```
GameState(board=[[2, 4], [2, 0]], score=8, gameWon=true)
```

Hinweis: Nutze `rotateRight`! Du kannst auch beide bisherigen `rotateRight`-Funktionen unverändert lassen und die zweite Funktion nutzen um die Aufgabe zu lösen.

1.8 Aufgabe

Ändere die Funktion `rotateMoveLeftRotateBack`. Dieser wird der Zustand des Spiels und ein Integer n übergeben. Sie gibt einen neuen Zustand zurück, nachdem das Spielfeld n mal nach rechts gedreht, einmal nach links gekippt, und wieder zurückgedreht wurde. Du kannst davon ausgehen, dass n kleiner als 4 ist.

```
rotateMoveLeftRotateBack(GameState(listOf(listOf(0, 2, 4),
                                          listOf(2, 1024, 1024),
                                          listOf(2, 4, 2)),
                                   14, false ), 2)
```

```
GameState(board=[[0, 2, 4], [0, 2, 2048], [2, 4, 2]], score=2062, gameWon=true)
```

```
rotateMoveLeftRotateBack(GameState(listOf(listOf(2, 2),
                                          listOf(0, 4)),
                                   100, false), 3)
```

```
GameState(board=[[2, 2], [0, 4]], score=100, gameWon=false)
```

Hinweis: Nutze `rotateRight` und `moveBoardLeft`!

1.9 Aufgabe

Ändere Funktion `move`. Der neuen Funktion wird der Zustand des Spiels und ein Character, der für eine Richtung steht(`w`, `a`, `s` oder `d`) übergeben. Sie gibt den neuen Spielzustand nach dem Kippen in diese Richtung zurück.

```
move(GameState(listOf(listOf(8, 2),
                      listOf(16, 2)),
               70, true),'s')
```

```
GameState(board=[[8, 0], [16, 4]], score=74, gameWon=true)
```

```
move(GameState(listOf(listOf(0, 2, 4),
                      listOf(2, 2, 2),
                      listOf(2, 4, 2)),
               30, false), 'd')
```

```
GameState(board=[[0, 2, 4], [0, 2, 4], [2, 4, 2]], score=34, gameWon=false)
```

Hinweis: Nutze `computeNecessaryRotationsBeforeMove` und `rotateMoveLeftRotateBack`!

1.10 Aufgabe

Ändere die Funktion `checkMovesPossible`. Dieser wird der Zustand des Spiels übergeben. Sie gibt zurück, ob das Spielfeld beim Kippen in eine Richtung verändert wird.

```
checkMovesPossible(GameState(listOf(listOf(8, 2),
                                    listOf(16, 2)),
                             20, false))
```

```
true
```

```
checkMovesPossible(GameState(listOf(listOf(2, 4, 8),
                                    listOf(4, 8, 4),
                                    listOf(8, 4, 2)),
                             50, true))
```

```
false
```

Hinweis: Nutze `rotateMoveLeftRotateBack`!

1.11 Aufgabe

Ändere die Funktion `update`. Dieser wird der Zustand des Spiels und ein Character, der für eine Richtung steht(`w`, `a`, `s` oder `d`) übergeben. Sie berechnet den neuen Zustand des Spiels nach dem Kippen in diese Richtung und dem Hinzufügen von zwei Elementen mit `addTwoOrFour`.

```
update(GameState(listOf(listOf(0, 2, 4),
                        listOf(2, 2, 2),
                        listOf(2, 4, 2)),
                 4, false),
        'd')
```

```
GameState(board=[[2, 2, 4], [2, 2, 4], [2, 4, 2]], score=8, gameWon=false)
```

**Nutze `move` und `addTwoOrFour`!

1.12 Aufgabe

Implementiere eine Funktion `printStrings`. Diese gibt eine Liste von Strings zeilenweise in der Konsole aus.

```
printStrings(listOf("akjefhkajsd", "adfsf", "12e124"))
```

```
akjefhkajsd
adfsf
12e124
```

```
printStrings(listOf("", "90798"))
```

```

90798
```

1.13 Aufgabe

Implementiere eine Funktion `constructStartGameState`. Dieser wird eine Zeile und eine Spaltenanzahl übergeben. Sie erzeugt den Anfangszustand. Das Spielbrett hat dabei die angegebenen Dimensionen.

```
constructStartGameState(2, 3)
```

```
GameState(board=[[2, 0, 0], [0, 2, 0]], score=0, gameWon=false)
```

```
constructStartGameState(4, 4)
```

```
GameState(board=[[2, 2, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]], score=0, gameWon=false)
```

Hinweis: Nutze `createStartBoard`!

1.14 Aufgabe

Verbinde die bereits geschriebenen Funktion zur Funktion `play2048HighScore`. Dieser Funktion werden die gewünschte Zeilen und Spaltenanzahl übergeben. Das Spiel kann mit der Taste `q` beendet werden. Wenn die Zahl 2048 auf dem Spielfeld ist, hat der Spieler gewonnen. Bei einem früheren Spielende verliert er das Spiel. Die Punktezahl wird in jeder Runde angezeigt.

```
|      0 |      0 |      0 |      0 |
|      0 |      0 |      2 |      0 |
|      0 |      2 |      0 |      0 |
|      0 |      0 |      0 |      0 |
Score: 0
w
|      0 |      2 |      2 |      0 |
|      0 |      0 |      0 |      0 |
|      0 |      0 |      0 |      0 |
|      2 |      0 |      0 |      0 |
Score: 0
a
|      4 |      0 |      0 |      2 |
|      0 |      0 |      0 |      0 |
|      0 |      0 |      0 |      0 |
|      2 |      0 |      0 |      0 |
Score: 4
a
|      4 |      2 |      0 |      0 |
|      0 |      0 |      0 |      0 |
|      0 |      0 |      0 |      0 |
|      2 |      0 |      0 |      2 |
Score: 4
d
|      2 |      0 |      4 |      2 |
|      0 |      0 |      0 |      0 |
|      0 |      0 |      0 |      0 |
|      0 |      0 |      0 |      4 |
Score: 8
q
Du hast verloren
```

Hinweis: Nutze `constructStartGameState`, `checkMovesPossible`, `showGameState`, `printStrings` `readChar`, `update` und `showResult`!