

# awari

Jon Sparring

October 18, 2018

## 1 Lærervejledningn

**Emne** Typer, lister, mønstergenkendelse, funktionsprogrammering

**Sværhedsgrad** Hård

## 2 Introduktion

I denne opgave skal I programmere spillet Awari, som er en variant af Kalaha. Awari er et gammelt spil fra Afrika, som spilles af 2 spillere, med 7 pinde og 36 bønner. Pindene lægges så der dannes 14 felter ('pits' på engelsk), hvoraf 2 er hjemmefelter. Bønnerne fordeles ved spillet start med 3 i hvert felt på nær i hjemmefelterne. Startopstillingen er illustreret i Figur 1.

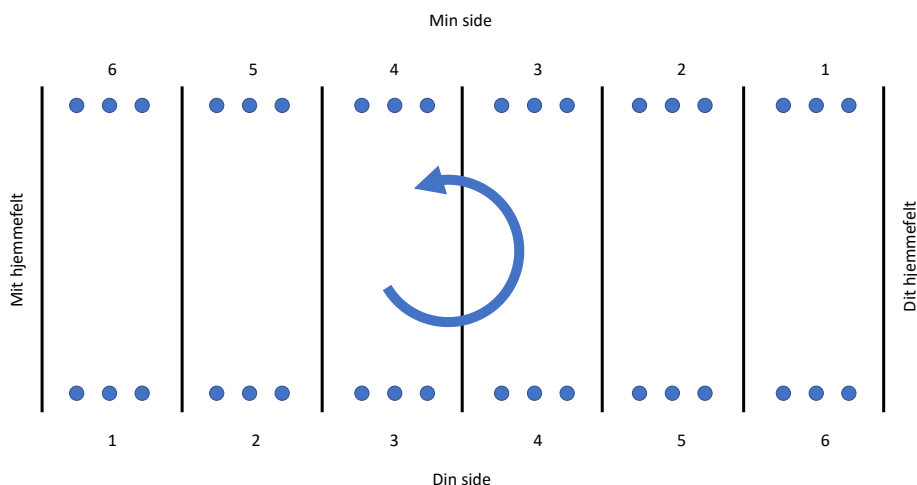


Figure 1: Udgangsstillingen for spillet Awari.

Spillerne skiftes til at spille en tur efter følgende regler:

- En tur spilles ved at spilleren tager alle bønnerne i et af spillerens felter 1-6 og placerer dem i de efterfølgende felter inkl. hjemmefelterne en ad gangen og mod uret. F.eks., kan første spiller vælge at tage bønnerne fra felt 4, hvorefter spilleren skal placere en bønne i hver af felterne 5, 6 og hjemmefeltet.
- Hvis sidste bønne lægges i spillerens hjemmefelt, får spilleren en tur til.
- Hvis sidste bønne lander i et tom felt som ikke er et hjemmefelt, og feltet overfor indeholder bønner, så flyttes sidste bønne til spillerens hjemmefelt, og alle bønnerne overfor fanges og flyttes ligeså til hjemmefeltet.
- Spillet er slut når en af spillerne ingen bønner har i sine felter 1-6, og vinderen er den spiller, som har flest bønner i sit hjemmefelt.

### 3 Opgave(r)

1. I skal implementere spillet Awari, som kan spilles af 2 spillere, og skrive en kort rapport. Kravene til jeres aflevering er:
  - Koden skal organiseres som bibliotek, en applikation og en test-applikation.
  - Biblioteket skal tage udgangspunkt i følgende signatur- og implementationsfiler:

**Listing 1 awariLibIncompleteLowComments.fsi:**  
**En ikke færdigskrevet signaturfil.**

```
1 module Awari
2 type pit = // intentionally left empty
3 type board = // intentionally left empty
4 type player = Player1 | Player2
5
6 /// Print the board
7 val printBoard : b:board -> unit
8
9 /// Check whether a pit is the player's home
10 val isHome : b:board -> p:player -> i:pit -> bool
11
12 /// Check whether the game is over
13 val isGameOver : b:board -> bool
14
15 /// Get the pit of next move from the user
16 val getMove : b:board -> p:player -> q:string -> pit
17
18 /// Distributing beans counter clockwise,
19 /// capturing when relevant
20 val distribute :
21     b:board -> p:player -> i:pit -> board * player * pit
22
23 /// Interact with the user through getMove to perform
24 /// a possibly repeated turn of a player
25 val turn : b:board -> p:player -> board
26
27 /// Play game until one side is empty
28 val play : b:board -> p:player -> board
```

**Listing 2 awariLibIncomplete.fs:**  
**En ikke færdigskrevet implementationsfil.**

```
1 module Awari
2 type pit = // intentionally left empty
3 type board = // intentionally left empty
4 type player = Player1 | Player2
5
6 // intentionally many missing implementations and additions
7
8 let turn (b : board) (p : player) : board =
9     let rec repeat (b: board) (p: player) (n: int) : board =
10         printBoard b
11         let str =
12             if n = 0 then
13                 sprintf "Player %A's move? " p
14             else
15                 "Again? "
16         let i = getMove b p str
17         let (newB, finalPitsPlayer, finalPit) = distribute b p i
18         if not (isHome b finalPitsPlayer finalPit)
19             || (isGameOver b) then
20             newB
21         else
22             repeat newB p (n + 1)
23     repeat b p 0
24
25 let rec play (b : board) (p : player) : board =
26     if isGameOver b then
27         b
28     else
29         let newB = turn b p
30         let nextP =
31             if p = Player1 then
32                 Player2
33             else
34                 Player1
35         play newB nextP
```

En version af signaturfilen med yderligere dokumentation og implementationsfilen findes i Absalon i opgaveområdet for denne opgave.

- Jeres løsning skal benytte funktionsparadigmet såvidt muligt.
- Koden skal dokumenteres vha. kommentarstandard for F#
- Jeres aflevering skal indeholde en afprøvning efter white-box metoden.
- I skal skrive en kort rapport i LaTeX på maks. 10 sider og som indeholder:
  - en beskrivelse af jeres design og implementation
  - en gennemgang af jeres white-box afprøvning
  - kildekoden som appendiks.