

Programmering og Problemløsning

Datalogisk Institut, Københavns Universitet

Arbejdsseddel 7 - gruppeopgave

Jon Sparring

21. oktober - 10. november.
Afleveringsfrist: søndag d. 10. november kl. 22:00.

Emnerne for denne arbejdsseddel er:

- rekursion,
- pattern matching,
- sumtyper,
- endelige træer.

Opgaverne er delt i øve- og afleveringsopgaver. I denne periode skal I arbejde i grupper med jeres afleveringsopgaver. Regler for gruppe- og individuelle afleveringsopgaver er beskrevet i "Noter, links, software m.m." → "Generel information om opgaver".

Øveopgaver

- 7ø.0 Omskriv funktionen `insert`, som benyttes i forbindelse med funktionen `isort` (insertion sort) fra forelæsningen, således at den benytter sig af pattern matching på lister.
- 7ø.1 Omskriv funktionen `bsort` (bubble sort) fra forelæsningen således at den benytter sig af pattern matching på lister. Funktionen kan passende benytte sig af "nested pattern matching" i den forstand at den kan implementeres med et match case der udtrækker de to første elementer af listen samt halen efter disse to elementer.
- 7ø.2 Opskriv black-box tests for de to sorteringsfunktioner og vær sikker på at grænsetilfældene dækkes (ingen elementer, et element, to elementer, samt flere elementer, sorteret, omvendt sorteret, etc.)
- 7ø.3 Omskriv funktionen `merge`, som benyttes i forbindelse med funktionen `msort` (mergesort) fra forelæsningen, således at den benytter sig af pattern matching på lister.

7ø.4 Opskriv black-box tests for sorteringsfunktionen msort og vær sikker på at grænsetilfældene dækkes (ingen elementer, et element, to elementer, samt flere elementer, sorteret, omvendt sorteret, etc.)

7ø.5 Lav en funktion dayToNumber : weekday -> int, der givet en ugedag returnerer et tal, hvor mandag skal give tallet 1, tirsdag tallet 2 osv.

7ø.6 Lav en funktion nextDay : weekday -> weekday, der givet en ugedag returnerer den næste dag, så mandag skal give tirsdag, tirsdag skal give onsdag, osv, og søndag skal give mandag.

7ø.7 Givet typen for ugedage øverst på denne ugeseddel, lav en funktion numberToDay : int -> weekday option, sådan at numberToDay n returnerer None, hvis n ikke ligger i intervallet 1...7, og returnerer Some d , hvor d er den til n hørende ugedag, hvis n ligger i intervallet 1...7.

Det skulle gerne gælde, at numberToDay (dayToNumber d) \rightsquigarrow Some d for alle ugedage d .

7ø.8 Ved at benytte biblioteket ImgUtil, som beskrevet i forelæsningen, er det muligt at tegne simpel liniegrafik samt fraktaler, som f.eks. Sierpinski-fraktalen, der kan tegnes ved at tegne små firkanter bestemt af et rekursivt mønster. Koden for Sierpinski-trekanten er givet som følger:

```
open ImgUtil

let rec triangle bmp len (x,y) =
  if len < 25 then setBox blue (x,y) (x+len,y+len) bmp
  else let half = len / 2
        do triangle bmp half (x+half/2,y)
        do triangle bmp half (x,y+half)
        do triangle bmp half (x+half,y+half)

do runSimpleApp "Sierpinski" 600 600 (fun bmp -> triangle bmp 512
  (30,30) |> ignore)
```

Tilpas funktionen således at trekanten tegnes med røde streger samt således at den kun tegnes ned til dybde 2 (hint: du skal ændre betingelsen len < 25).

7ø.9 I stedet for at benytte ImgUtil.runSimpleApp funktionen skal du nu benytte ImgUtil.runApp, som giver mulighed for at din løsning kan styres ved brug af tastaturet. Funktionen ImgUtil har følgende type:

```
val runApp : string -> int -> int
           -> (int -> int -> 's -> System.Drawing.Bitmap)
           -> ('s -> System.Windows.Forms.KeyEventArgs
               -> 's option)
           -> 's -> unit
```

De tre første argumenter til runApp er vinduets titel (en streng) samt vinduets initiale vidde og højde. Funktionen runApp er parametrisk over en brugerdefineret type af tilstande ('s). Antag at funktionen kaldes som følger:

```
runApp title width height draw react init
```

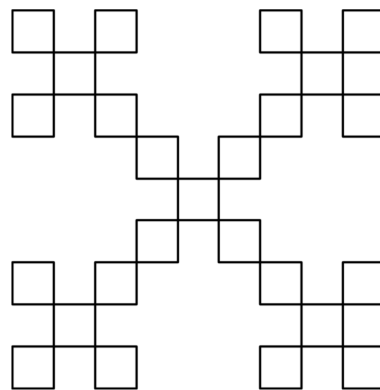
Dette kald vil starte en GUI applikation med titlen title, vidden width og højden height. Funktionen draw, som brugeren giver som 4. argument kaldes initielt når applikationen starter og

hver gang vinduets størrelse justeres eller ved at funktionen `react` er blevet kaldt efter en tast er trykket på tastaturet. Funktionen `draw` modtager også (udover værdier for den aktuelle vidde og højde) en værdi for den brugerdefinerede tilstand, som initielt er sat til værdien `init`. Funktionen skal returnere et bitmap, som for eksempel kan konstrueres med funktionen `ImgUtil.mk` og ændres med andre funktioner i `ImgUtil` (f.eks. `setPixel`).

Funktionen `react`, som brugeren giver som 5. argument kaldes hver gang brugeren trykker på en tast. Funktionen tager som argument en værdi svarende til den nuværende tilstand for applikationen samt et argument der kan benyttes til at afgøre hvilken tast der blev trykket på.¹ Funktionen kan nu (eventuelt) ændre på dens tilstand ved at returnere en ændret værdi for denne.

Tilpas applikationen således at dybden af fraktalen kan styres ved brug af piletasterne, repræsenteret ved værdierne `System.Windows.Forms.Keys.Up` og `System.Windows.Forms.Keys.Down`.

- 7ø.10 Med udgangspunkt i øvelsesopgave 7ø.8 skal du i denne opgave implementere en GUI-applikation der kan tegne en version af X-fraktalen som illustreret nedenfor (eventuelt i en dybde større end 2).



Bemærk at det ikke er et krav at dybden på fraktalen skal kunne styres med piletasterne som det er tilfældet med Sierpinski-fraktalen i øvelsesopgave 7ø.9.

Afleveringsopgaver

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.

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.

¹Hvis `e` har typen `System.Windows.Forms.KeyEventArgs` kan betingelsen `e.KeyCode == System.Windows.Forms.Keys.Up` benyttes til at afgøre om det var tasten "Up" der blev trykket på.

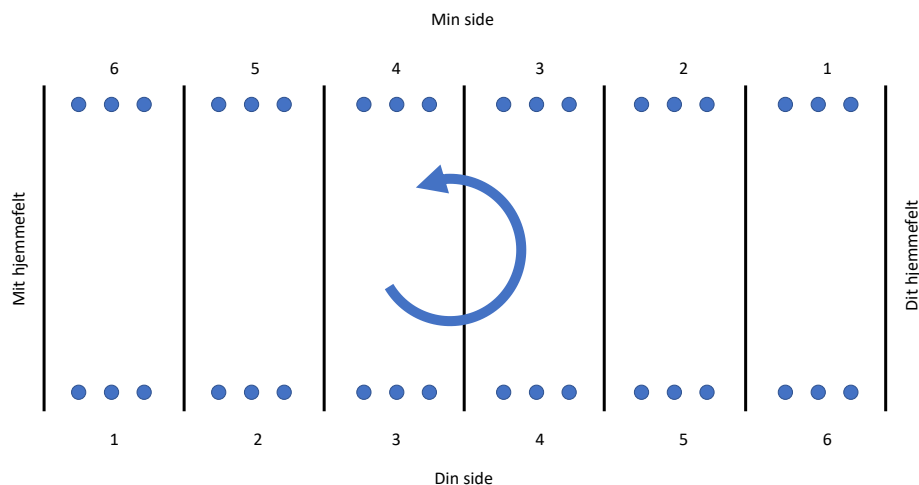


Figure 1: Udgangspstillingen for spillet Awari.

- 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.

7g.0 (a) 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
18     i
19     if not (isHome b finalPitsPlayer finalPit)
20       || (isGameOver b) then
21       newB
22     else
23       repeat newB p (n + 1)
24   repeat b p 0
25
26 let rec play (b : board) (p : player) : board =
27   if isGameOver b then
28     b
29   else
30     let newB = turn b p
31     let nextP =
32       if p = Player1 then
33         Player2
34       else
35         Player1
36     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. kommentarstandarden 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.

Afleveringen skal bestå af

- en zip-fil og
- en rapport i pdf-format.

Zip-filen skal indeholde mapperne `tex` og `src` og filen `README.txt`. Mapperne skal indeholde henholdsvis LaTeX koden m.m. og fsharp koden. Der skal være en fsharp tekstfil per fsharp-opgave, og de skal navngives `7g0.fsx` osv. Rapporten skal kunne oversættes vha. `pdflatex` kommandoen i `tex`-mappen, og fsharp koden skal kunne oversættes med `fsharpc` og køres med `mono` i `src` mappen. Filen `README.txt` skal ganske kort beskrive, hvordan rapporten oversættes til pdf og koden køres.

God fornøjelse.