Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 6 - gruppeopgave

Jon Sporring

21. oktober - 5. november.
Afleveringsfrist: lørdag d. 5. november kl. 22:00.

I denne periode skal vi arbejde med en lidt større programmeringsopgave og kigge på højereordens funktioner og abstrakte datatyper.

Denne arbejdsseddels læringsmål er:

- Kunne definere funktioner som tager andre funktioner som argument og/eller giver funktioner som returværdi
- Kunne definere funktioner ved currying
- Kunne arbejde med den abstrakte datatype træe.

Opgaverne er opdelt 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 (in English)

I det følgene skal I arbejde med polynomier. Et polynomium af grad n skrives som

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = \sum_{i=0}^{n} a_i x^i.$$

6ø0 Skriv en funktion `poly: float list -> float -> float`, der tager som argumenter (1) en liste `a` af koefficienter med `a.[i]` $= a_i$ og (2) en x-værdi for derefter at returnere polynomiets værdi. Afprøv funktionen ved at lave tabeller for et lille antal polynomier af forskellig grad med forskellige koefficienter og forskellige værdier for x, og validér den beregnede værdi.

6ø1 Definer en funktion `line : float -> float -> float -> float` ved brug af `poly`, således at `line a0 a1 x` beregner værdien for et 1. grads polynomium hvor `a0` $= a_0$, `a1` $= a_1$ og `x` $= x$. Afprøv funktionen ved at tabellere værdier for `line` med det samme sæt af koefficienter $a_0 \neq 0$ og $a_1 \neq 0$ og et passende antal værdier for x.

6ø2 Benyt partiel anvendelse af `line` til at definere en funktion `theLine : float -> float`, hvor parametrene `a0` og `a1` er sat til det samme som brugt i Opgave 6ø1. Afprøv `theLine` tilsvarende som `line` afprøves i Opgave 6ø1.

De følgende opgaver omhandler integration. Integralet af næsten alle integrable funktioner kan approximeres som

$$\int_a^b f(x)\,dx \simeq \sum_{i=0}^{n-1} f(x_i)\Delta x,$$

hvor $x_i = a + i\Delta x$ og $\Delta x = \frac{b-a}{n}$.

6ø3 Skriv en funktion `integrate : n:int -> a:float -> b:float -> (f : float -> float) -> float`, hvis argumenter `n`, `a`, `b`, er som i ligningerne, og `f: float -> float` er en F#-funktion, som repræsenterer en integrabel én-dimensionel funktion. Afprøv `integrate` på `theLine` fra Opgave 6ø2 og på `cos` med $a = 0$ og $b = \pi$. Udregn integralerne analytisk og sammenlign med resultatet af `integrate`.

6ø4 Funktionen `integrate` er en approximation, og præcisionen afhænger af n. Undersøg afhængigheden ved at udregne fejlen, dvs. forskellen mellem det analytiske resultat og approximationen for værdier af n. Dertil skal du lave to funktioner `integrateLine : n:int -> float` og `integrateCos : n:int -> float` vha. `integrate`, `theLine` og `cos`, hvor værdierne for a og b og f er fastlåste. Afprøv disse funktioner for $n = 1, 10, 100, 1000$. Overvej om der er en tendens i fejlen, og hvad den kan skyldes.

The following exercises are about expanding and using the following recursive sumtype, which can be used for modelling arithmetic expressions:

```
type expr = Const of int | Add of expr * expr | Mul of expr * expr
```

For eksempel, the expression $(5+8)*9$ is represented by

```
Mul (Add (Const 5, Const 8), Const 9)
```

6ø5 Implement a recursive function `eval : expr -> int` that takes an expression value as argument and returns the integer resulting from evaluating the expression term. The expression `eval (Mul (Add (Const 5, Const 8), Const 9))` should return the integer value `117`.

6ø6 Extend the type `expr` with cases for subtraction and division, think about the type of the evaluator extended expression language: What should be the result when dividing by zero? Modify and extend your evaluator to include subtraction and division, and test whether your implementation works in practice.

In the following exercises, we shall investigate the following recursive type definition for binary trees:

```
type 'a tree = Leaf of 'a | Tree of 'a tree * 'a tree
```

A term of the form `Leaf $x$` is called a *leaf* (or *leaf node*); a term of the form `Tree $(t1, t2)$` is called a *tree node*. The tree type is generic in the type of data that in the leaf nodes.

6ø7 Write a function `leafs : 'a tree -> int` that returns the number of leaf nodes appearing in a tree. Evaluate that your function works as expected.

6ø8 Write a function `sum : int tree -> int` that returns the sum of the integer values appearing in the leafs of the tree. Evaluate that your function works as expected.

6ø9 Write a function `preorder : 'a tree -> 'a list` that returns a tree's leaves in *preorder*. The leaves are listed in preorder, if all the leaves of `t1` in `Tree (t1, t2)` are listed before the leaves of `t2`, and this is the case for all tree nodes in a tree. For example, the leaves of `Tree (Tree (Leaf 5, Leaf 8), Leaf 3)` in preorder are `[5; 8; 3]`.

Test whether your function works as specified.

6ø10 Write a function

```
find : pred: ('a -> bool) -> 'a tree -> 'a option
```

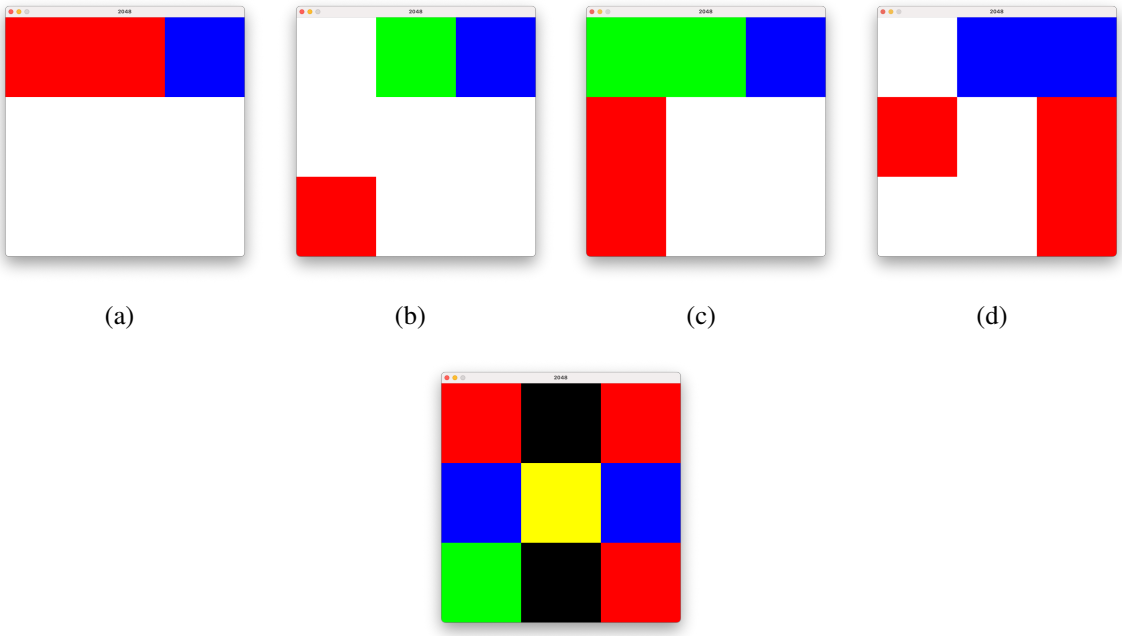
that returns the first value in preorder that satisfies the provided predicate `pred`. If no such value appears in the tree, the function should return the value `None`.

Test whether your function works as specified.

Afleveringsopgaver (in English)

2048 is a popular solitaire board game available, e.g., online on `https://2048.io/`. In this exercise, you are to implement a version of this game in F# and using `Canvas`. The rules, you are to implement are as follows:

1. The board is a square board with 3×3 field.
2. The pieces are colored squares: red, green, blue, yellow, black corresponding to the values 2, 4, 8, 16, 32.
3. There can at most be one piece per field on the board.



(a) (b) (c) (d)

(e)

Figure 1: Some examples

4. The initial conditions is a red and a blue piece placed on the board.
5. The game can be tilted left, right, up, down by pressing the corresponding arrow keys.
6. When the game is tilted, then all the pieces are to be moved to the corresponding side of the board.
7. If two pieces of the same color are pushed into each other in the process of tilting, then they are replaced by a single piece of double the value. For example, the board in Figure 1(a) is tilted to the right, and the two red pieces are replaced with green piece. However, replacement is not performed in a cascading fashion, e.g., the board in Figure 1(c) is tilted to the right combining the two green to a blue, but the resulting two blues are not combined.
8. Two black pieces are combined into one black piece.
9. After each turn, a new red piece is to be placed randomly on an available field on the board.
10. The game ends when there are no possible moves and no empty locations for a new piece to spawn.

In your solution, you are to represented a board with its pieces as a list of pieces, where each piece has a color and a position. This is captured by the following type abbreviations:

```
type pos = int*int // A 2-dimensional vector in board-coordinats (not
   pixels)
type value = Red | Green | Blue | Yellow | Black // piece values
type piece = value*pos //
type state = piece list // the board is a set of randomly organized
   pieces
```

In the following, the first coordinate in pos will thought of as a up-down axis also called the row, and the second as an left-right axis also called the column with $(0,0)$ being the top-left.

6g0 Make a library consisting of a signature and an implementation file. The library must contain the following functions

```
// convert a 2048-value v to a canvas color E.g.,
// > fromValue Green;;
// val it: color = { r = 0uy
//   g = 255uy
//   b = 0uy
//   a = 255uy }
fromValue: v: value -> Canvas.color

// give the 2048-value which is the next in order from c, e.g.,
// > nextColor Blue;;
// val it: value = Yellow
// > nextColor Black;;
// val it: value = Black
nextColor: c: value -> value

// return the list of pieces on a column k on board s, e.g.,
// > filter 0 [(Blue, (1, 0)); (Red, (0, 0))];;
// val it: state = [(Blue, (1, 0)); (Red, (0, 0))]
// > filter 1 [(Blue, (1, 0)); (Red, (0, 0))];;
// val it: state = []
filter: k: int -> s: state -> state

// tilt all pieces on the board s to the left (towards zero on
// the first coordinate), e.g.,
// > shiftUp [(Blue, (1, 0)); (Red, (2, 0)); (Black, (1,1))];;
// val it: state = [(Blue, (0, 0)); (Red, (1, 0)); (Black, (0,
   1))]
shiftUp: s: state -> state

// flip the board s such that all pieces position change as
// (i,j) -> (2-i,j), e.g.
// > flipUD [(Blue, (1, 0)); (Red, (2, 0))];;
// val it: state = [(Blue, (1, 0)); (Red, (0, 0))]
flipUD: s: state -> state

// transpose the pieces on the board s such all piece positiosn
// change as (i,j) -> (j,i), e.g.,
// > transpose [(Blue, (1, 0)); (Red, (2, 0))];;
// val it: state = [(Blue, (0, 1)); (Red, (0, 2))]
transpose: s: state -> state

// find the list of empty positions on the board s, e.g.,
// > empty [(Blue, (1, 0)); (Red, (2, 0))];;
// val it: pos list = [(0, 0); (0, 1); (0, 2); (1, 1); (1, 2);
   (2, 1); (2, 2)]
empty: s: state -> pos list
```

```
// randomly place a new piece of color c on an empty position on
// the board s, e.g.,
// > addRandom Red [(Blue, (1, 0)); (Red, (2, 0))];;
// val it: state option = Some [(Red, (0, 2)); (Blue, (1, 0));
   (Red, (2, 0))]
addRandom: c: value -> s: state -> state option
```

With these functions and Canvas it is possible to program the game in a few lines. Add the following to your library:

(a) Write a canvas draw function

```
draw: w: int -> h: int -> s: state -> canvas
```

which makes a new canvas and draws the board in s.

(b) Write a canvas react function

```
react: s: state -> k: key -> state option
```

which titles the board base according to the arrow-key, the user presses. Note that tilt left is given by the `shiftLeft` function. Tilt right can be accomplished by `fliplr >> shiftLeft >> fliplr`, and tilt up and down can likewise be accomplished with the additional use of `transpose`.

Finally, make an application program, which calls `runApp "2048" 600 600 draw react board`.

All above mentioned functions are to be documented using the XML-standard, and simple test examples are to be made for each function showing that it likely works.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `6g.zip`
- en opgavebesvarelse i pdf-format.

Zip-filen skal indeholde:

- filen `README.txt` som er en textfil med jeres navne og dato arbejdet, samt en beskrivelse af, hvordan man kører jeres kode.
- en `src` mappe med følgende og kun følgende filer:

  6g0Lib.fsi, 6g0Lib.fs, 6g0App.fsx

  Funktionerne skal være dokumenteret ifølge dokumentationsstandarden ved brug af `<summary>`, `<param>` og `<returns>` XML tagsne.

- pdf-dokumentet skal være lavet med LaTeX, benytte `rapport.tex` skabelonen, skal dokumentere jes løsning og indeholde figurer, der viser outputgrafik fra canvas for opgaverne.

God fornøjelse.