Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 5 - individuel opgave

Jon Sporring

30. september - 4. oktober.
Afleveringsfrist: lørdag d. 5. oktober kl. 23:59.

Emnerne for denne arbejdsseddel er:

- lister,
- arrays.

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

Øveopgaver

5ø.0 Skriv en funktion `oneToN : n:int -> int list`, som returnerer listen af heltal `[1; 2; ...; n]`.

5ø.1 Skriv en funktion `multiplicity : x:int -> xs:int list -> int`, som tæller antallet af gange tallet `x` optræder i listen `xs`.

5ø.2 Skriv funktionen `split : xs:int list -> (xs1: int list) * (xs2: int list)`, som deler listen `xs` i 2 og returnerer resultatet som en tuple, hvor alle elementer med lige index er i første element og resten i andet element. F.eks. `split [x0; x1; x2; x3; x4]` skal returnere `([x0; x2; x4], [x1; x3;])`.

5ø.3 Definer en funktion `reverseApply : 'a -> ('a -> 'b) -> 'b`, sådan at kaldet `reverseApply x f` returnerer resultatet af funktionsanvendelsen `f x`.

5ø.4 Forklar forskellen mellem typerne `int -> (int -> int)` og `(int -> int) -> int`, og giv et eksempel på en funktion af hver type.

5ø.5 Brug `List.filter` til at lave en funktion `evens : int list -> int list`, der returnerer de lige heltal i en liste.

5ø.6 Brug `List.map` og `reverseApply` (fra Opgave 5ø.3) til at lave en funktion `applylist : ('a -> 'b) list -> 'a -> 'b list`, der anvender en liste af funktioner på samme element for at returnere en liste af resultater.

5ø.7 Opskriv funktionstyperne for `List.filter` og `List.foldBack`.

5ø.8 En snedig programmør definerer en sorteringsfunktion med definitionen `ssort xs = Set.toList (Set.ofList xs)`. For eksempel giver `ssort [4; 3; 7; 2]` resultatet `[2; 3; 4; 7]`. Diskutér, om programmøren faktisk er så snedig, som han tror.

5ø.9 Brug `Array.init` til at lave en funktion `squares: int -> int []`, sådan at kaldet `squares n` returnerer listen af de *n* første kvadrattal. For eksempel skal `squares 5` returnere arrayet `[|1; 4; 9; 16; 25|]`.

5ø.10 Skriv en funktion`reverseArray : 'a [] -> 'a []` ved brug af `Array.init` og `Array.length`, og som returnerer arrayet med elementerne i omvendt rækkefølge. For eksempel skal kaldet `printfn "%A" (reverseArray [|1..5|])` udskrive `[|5; 4; 3; 2; 1|]`.

5ø.11 Brug en while-løkke og overskrivning af array-elementer til at skrive en funktion `reverseArrayD : 'a [] -> unit`, som destruktivt opdaterer et array, så elementerne kommer i omvendt rækkefølge. Sekvensen

```
let aa = [|1..5|]
reverseArrayD aa
printfn "%A" aa
```

skal altså udskrive `[|5; 4; 3; 2; 1|]`.

5ø.12 Brug `Array2D.init`, `Array2D.length1` og `Array2D.length2` til at lave en funktion `transpose : 'a [,] -> 'a [,]` som returnerer det transponerede argument, dvs. spejler det over diagonalen.

Afleveringsopgaver

5i.0 En tabel kan repræsenteres som en ikke tom liste af lister, hvor alle listerne er lige lange. Listen `[[1; 2; 3]; [4; 5; 6]]` repræsenterer for eksempel tabellen

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

(a) Lav en funktion `isTable : 'a list list -> bool`, der givet en liste af lister afgør, om det er en lovlig ikke-tom tabel, altså om alle listerne har ens længde, og at der er mindst en liste med mindst et element.

(b) Lav en funktion `firstColumn : 'a list list -> 'a list`, der tager en liste af lister og returnerer listen af førsteelementer i de indre lister. F.eks. skal `firstColumn [[1; 2; 3]; [4; 5; 6]]` returnere listen `[1; 4]`. Hvis en eller flere af listerne er tomme, er `firstColumn` udefineret.

(c) Lav en funktion `dropFirstColumn : 'a list list -> 'a list list`, der tager en liste af lister og returnerer en liste af lister, hvor førsteelementerne i de indre lister er fjernet. F.eks. skal `dropFirstColumn [[1; 2; 3]; [4; 5; 6]]` returnere `[[2; 3]; [5; 6]]`.

(d) Lav en funktion `transpose : 'a list list -> 'a list list`, der spejler tabellens ingange over diagonalen, så den transponerede tabel til den herover viste tabel er

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Kaldet `transpose [[1; 2; 3]; [4; 5; 6]]` skal altså returnere `[[1; 4]; [2; 5]; [3; 6]]`. Det kan antages, at argumentet til `transpose` er en lovlig tabel, så advarsler om ufuldstændige mønstre er acceptable – hvis funktionen ellers virker. Bemærk, at `transpose (transpose t) = t`, hvis `t` er en tabel. Vink: Brug funktionerne `firstColumn` og `dropFirstColumn`.

5i.1 Brug funktionerne opremset i [Kapitel 11, Sporring] til at definere en funktion `concat : 'a list list -> 'a list`, der sammensætter en liste af lister til en enkelt liste. F.eks. skal `concat [[2]; [6; 4]; [1]]` give resultatet `[2; 6; 4; 1]`.

5i.2 Brug funktionerne fra [Kapitel 11, Sporring] til at definere en funktion `gennemsnit : float list -> float option`, der finder gennemsnittet af en liste af kommatal, såfremt dette er veldefineret, og `None`, hvis ikke.

Afleveringen skal bestå af

- en zip-fil

Zip-filen skal indeholde en `src` mappe og filen `README.txt`. Mappen skal indeholde fsharp koden, der skal være en fsharp tekstfil per fsharp-opgave, og de skal navngives `5i0.fsx` osv. De skal kunne oversættes med fsharpc, og de oversattte filer skal kunne køres med mono. Funktioner skal dokumenteres ifølge dokumentationsstandarden, og udover selve programteksten skal besvarelserne indtastes som kommentarer i de fsx-filer, de hører til. Filen `README.txt` skal ganske kort beskrive, hvordan koden køres.

God fornøjelse.