

Programmering og Problemløsning

Datalogisk Institut, Københavns Universitet

Arbejdsseddel 5 - individuel opgave

Jon Sparring

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

- 5ø0 Skriv en funktion `oneToN : n:int -> int list`, som returnerer listen af heltal `[1; 2; ...; n]`.
- 5ø1 Skriv en funktion `multiplicity: x:int -> xs:int list -> int`, som tæller antallet af gange tallet `x` optræder i listen `xs`.
- 5ø2 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])`.
- 5ø3 Definer en funktion `reverseApply : x:'a -> f:('a -> 'b) -> 'b`, sådan at kaldet `reverseApply x f` returnerer resultatet af funktionsanvendelsen `f x`.
- 5ø4 Forklar forskellen mellem typerne `int -> (int -> int)` og `(int -> int) -> int`, og giv et eksempel på en funktion af hver type.
- 5ø5 Brug `List.filter` til at lave en funktion `evens : lst:int list -> int list`, der returnerer de lige heltal i liste `lst`.

- 5ø6 Brug `List.map` og `reverseApply` (fra Opgave 5ø3) til at lave en funktion `applylist : list:(a -> b) list -> x:a -> b list`, der anvender en liste af funktioner `lst` på samme element `x` for at returnere en liste af resultater.
- 5ø7 Opskriv typerne for funktionerne `List.filter` og `List.foldBack`.
- 5ø8 Brug `Array.init` til at lave en funktion `squares: n:int -> int []`, sådan at kaldet `squares n` returnerer arrayet af de n første kvadrattal. For eksempel skal `squares 5` returnere arrayet `[1; 4; 9; 16; 25]`.
- 5ø9 Skriv en funktion `reverseArray : arr:a [] -> a []` ved brug af `Array.init` og `Array.length`, og som returnerer arrayet med elementerne i omvendt rækkefølge af `arr`. For eksempel skal kaldet `printfn "%A" (reverseArray [|1..5|])` udskrive `[5; 4; 3; 2; 1]`.
- 5ø10 Brug en `while`-løkke og overskrivning af array-elementer til at skrive en funktion `reverseArrayD : arr:a [] -> unit`, som overskriver værdierne i arrayet `arr`, så elementerne kommer i omvendt rækkefølge. Sekvensen
- ```
let aa = [|1..5|]
reverseArrayD aa
printfn "%A" aa
```
- skal altså udskrive `[5; 4; 3; 2; 1]`.
- 5ø11 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

- 5i0 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 : llst:a list list -> bool`, der givet en liste af lister afgør, om det er en lovlig ikke-tom tabel. For at det er en lovlig ikke-tom tabel, skal der gælde følgende:
- Der er mindst en liste med mindst et element.
  - Alle lister i tabellen har ens længde.
- (b) Lav en funktion `firstColumn : llst: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, skal funktionen returnere den tomme liste af heltal `[] : int list`.
- (c) Lav en funktion `dropFirstColumn : llst: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 : list:'a list list -> 'a list list`, der spejler tabel-  
lens indgange 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]]`. Bemærk, at `transpose (transpose t) = t`, hvis `t` er en tabel. Tip: Brug funktionerne `firstColumn` og `dropFirstColumn`.

5i1 Brug funktionerne opremset i [Kapitel 11, Spørring] 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]`.

Afleveringen skal bestå af

- en zip-fil, der hedder `5i_<navn>.zip` (f.eks. `5i_jon.zip`)

Zip-filen `5i_<navn>.zip` skal indeholde en `src` mappe og filen `README.txt`. I `src` skal der ligge følgende og kun følgende filer: `5i0.fsx`, `5i1.fsx` og `5i2.fsx` svarende til hver af delopgaverne. De skal kunne oversættes med `fsharpc`, og de oversatte filer skal kunne køres med `mono`. Funktioner skal dokumenteres ifølge dokumentationsstandarden som minimum ved brug af `<summary>`, `<param>` og `<returns>` XML-tagsne. Udover selve koden skal besvarelser indtastes som kommentarer i de `fsx`-filer, de hører til. Filen `README.txt` skal ganske kort beskrive, hvordan koden oversættes og køres.

God fornøjelse.