Learning to Program with F#
Exercises
Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

August 24, 2020

0.1 Lists

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

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

0.1.3: 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])`.

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

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

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

0.1.7: Brug `List.map` og `reverseApply` (fra Opgave 4) til at lave en funktion `applylist : lst:('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.

0.1.8: Opskriv typerne for funktionerne `List.filter` og `List.foldBack`.

0.1.9: 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.

0.1.10: 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|]`.

0.1.11: 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|]`.

0.1.12: 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|]`.

0.1.13: 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.

0.1.14: 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 : llst:'a list list -> 'a list list`, der spejler tabellens 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`.

0.1.15: 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]`.

0.1.16: 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.