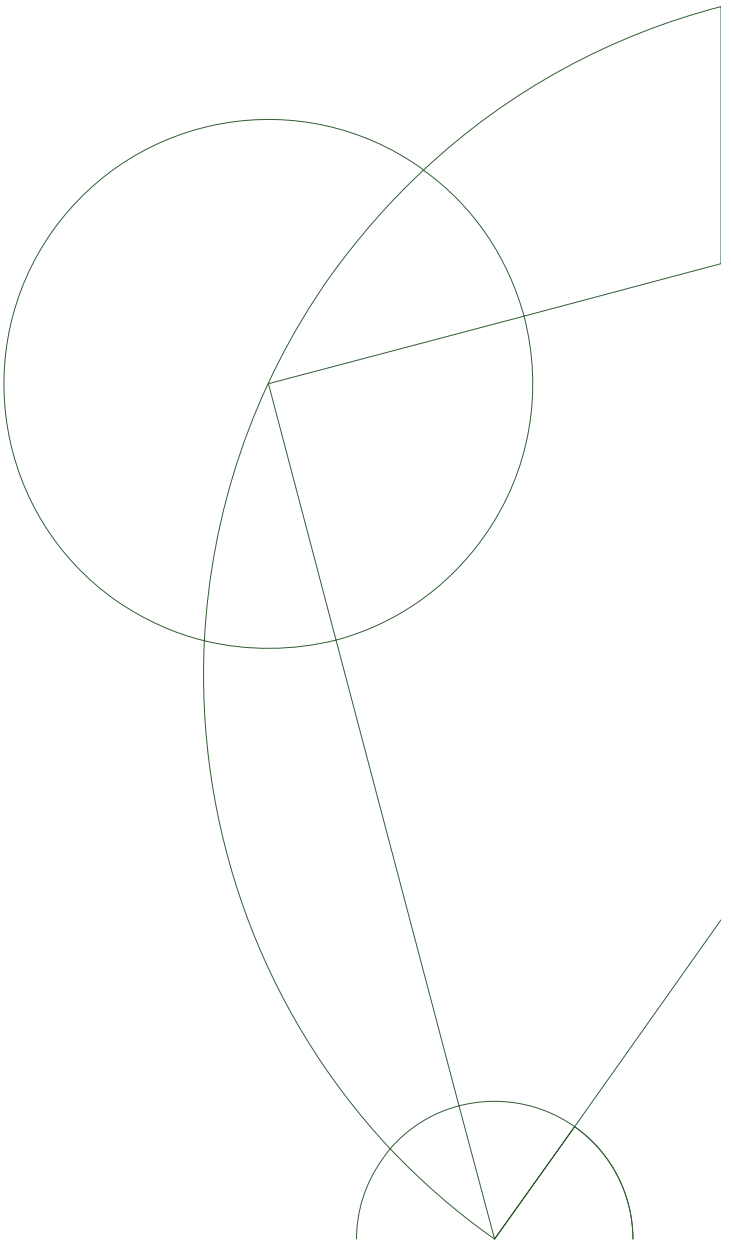
Programmering og Problemløsning

Aflevering 5i

Adam Ingwersen

Datalogisk Institut
Københavns Universitet

January 27, 2017



1 Programbeskrivelse

1.1 5i.1)

For at folde nestede lister ud, anvendes `List.foldBack` samt append-operatoren for lister:

```
let concat list = List.foldBack(fun x y -> x@y) list []
```

1.2 5i.2)

Denne delopgave at returnere et gennemsnit af en liste af `option`-typen. Til dette formål, er der her valgt, at anvende en hjælpefunktion, `gennemsnitHelp`:

```
let gennemsnitHelp liste =
 liste
  > List.fold (fun (elem, acc) x -> (elem + x, acc + 1.0))
     (0.0, 0.0)
  > (fun (elem, acc) -> elem / acc)
```

..For derefter, at konstruere funktionen, der anvender option-typen:

```
let gennemsnit liste =
 try
  let avg = gennemsnitHelp liste
  Some avg
 with _ -> None
```

1.3 5i.3)

I denne delopgave skal vi sortere et array funktionelt. Denne tilgang er inspireret af mergesort. Her har det været smart at tage en approach, der bygger på tre funktioner, til at løse problemstillingen. Funktionerne `arraySplit, arrayMerge, arraySort` gør netop dette.

Funktionerne gør nogenlunde hvad, man skulle tro, de gør:

```
let rec arraySplit (arrIn: 'a []) (left: 'a []) (right: 'a
    []) =
 match arrIn with
   [ ] -> (left, right)
   _ -> (arraySplit (Array.tail arrIn) right (Array.append
     [ Array.head arrIn ] left))
```

I `arraySplit`, splittes et array op i to.

```
let rec arrayMerge ((left: 'a []), (right: 'a [])) =
 match (left, right) with
   (left, [ ]) -> left
   ([ ], right) -> right
   (_,_) -> if ((left.[0]) < (right.[0]))
                      then Array.append [ left.[0] ] (
                          arrayMerge (left.[1..], right))
              else Array.append [ right.[0] ] (arrayMerge (
                  left, right.[1..]))
```

I `arrayMerge` samles to arrays og sorteres i processen.

```
let rec arraySort = function
    [ ] -> [ ]
    [ x ] -> [ x ]
   testArr ->
  let (temp1, temp2) = arraySplit testArr [ ] [ ]
  arrayMerge ((arraySort temp1), arraySort temp2)
```

I `arraySort` samles de to ovenstående funktioner, for rekursivt at applikere et split og et merge på et vilkårligt, usorteret array. En umiddelbar kritik af denne løsning vil være, at der kun bliver splittet med eet element af gangen, og dette kan gøres med færre operationer. I størstedelen af praktiske implementeringer af denne slags funktioner, vil man som oftest være bedst tjent ved at bero sig på de indbyggede moduler - i dette tilfælde `List.sort`.

1.4 5i.4)

I denne delopgave skal en sorteringsfunktion implementeres imperativt - inspireret af insertion-sort. Indledningsvist konstrueres en hjælpefunktion, der kan bytte plads mellem to nærliggende (adjacent) elemeter i et array. Denne funktion, er blevet kaldt `swap`. Herefter laves en funktion `arraySortD`, der anvender `swap` i et for-loop til at bytte om på elementer, hvis disses værdier er sorteret i aftagende rækkefølge, de to elementer imellem. Der føres regnskab med med variablen `check` med det formål at gentage loopet, over hele array'et indtil slut. Hvis dette regnskab ikke blev ført, ville hele array'et blive traverseret en gang - men et array, eks: [3;5;2] ville blive til [3;2;5] - dette løses med `check` og rekursion.

```
let arraySortD (arr: 'a []) =
  let rec loop(arr: 'a []) =
    let mutable check = 0
    for i = 0 to ((Array.length arr) - 2) do
      if arr.[i] > arr.[i+1] then
        swap i arr
        check <- check + 1
    if check > 0 then loop arr else arr
  loop arr
```