Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 3 - individuel opgave

Jon Sporring

23. september - 1. oktober.
Afleveringsfrist: lørdag d. 1. oktober kl. 22:00.

I denne periode skal vi arbejde med lister. Lister er den første af en række abstrakte datastrukturer, vi skal kigge på, og som er velunderstøttet i F#. Det er så vigtig en datastruktur at F# både har syntaks der direkte understøtter lister og et bibliotek (modul) med mange ekstra funktioner til listebearbejdning, `List`.

Denne arbejdsseddels læringsmål er:

- at kunne arbejde med anonyme funktioner,
- at kunne oprette, gennemløbe og lave beregninger med lists vha. `List`-modulet,
- at kunne skrive rekursive funktioner, som tager lister som argument og som giver lister som returværdi,

Opgaverne er opdelt 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 (in English)

3ø0 In the following, you are to work with different ways to create a list:

(a) Make an empty list, and bind it with the name `lst`.

(b) Create a second list `lst2`, with `lst` and the cons operator `::`, which contains the single element `"F#"`. Consider whether the types of the old and new list are the same.

(c) Create a third list `lst3` which consists of 3 identical elements `"Hello"`, and which is created with `List.init` and the anonymous function `fun i -> "Hello"`.

(d) Create a fourth list `lst4` which is a concatenation of `lst2` and `lst3` using "`@`".

(e) Create a fifth list `lst5` as `[1; 2; 3]` using `List.init`

(f) Write a recursive function `oneToN : n:int -> int list` which uses the concatenation operator, "`@`", and returns the list of integers `[1; 2; ...; n]`. Consider whether it would be easy to create this list using the "`::`" operator.

(g) Write a recursive function `oneToNRev : n:int -> int list` which uses the cons operator, "`::`", and returns the list of integers `[n; ...; 2; 1]`. Consider whether it would be easy to create this list using the "`@`" operator.

3ø1 Use `List.map` write a function, which takes a list of integers and returns the list of floats where each element has been divided by `2.0`. For example, if the function is given the input `[1; 2; 3]`, then it should return `[0.5; 1.0; 1.5]`.

3ø2 Write a recursive function `rev: 'a list -> 'a list`, which uses the cons operator `"::"` to reverse the elements in a list.

3ø3 Write the types for the functions `List.filter` and `List.foldBack`.

3ø4 Make a function `avg: (lst: float list) -> float` using `List.fold` and `lst.Length` which calculates the average value of the elements of `lst`.

Afleveringsopgaver (in English)

In the following we are going to work with lists and Canvas. The module Canvas has the ability to perform simple turtle graphics. To draw in turtle graphics, we command a little invisible turtle, which moves on the canvas with a pen. The function `turtleDraw` is given a list of `turtleCmds`, such as `PenUp` and `PenDown` to raise and lower the pen, `Turn 250` and `Move 100` to turn 250 degrees and move 100 pixels, and `SetColor red` to pick a red pen. For example, the following example draws a fractal tree starting and ending in the center of the canvas:

```
#r "nuget:diku.canvas, 1.0.1"
open Canvas

let rec tree sz =
   if sz < 5 then
     [Move sz; PenUp; Move (-sz); PenDown]
   else
     [Move (sz/3); Turn -30]
     @ tree (sz*2/3)
     @ [Turn 30; Move (sz/6); Turn 25]
     @ tree (sz/2)
     @ [Turn -25; Move (sz/3); Turn 25]
     @ tree (sz/2)
     @ [Turn -25; Move (sz/6); PenUp; Move (-sz/3); Move (-sz/6);
   Move (-sz/3); Move (-sz/6); PenDown]

let w = 600
let h = w
let sz = 100

turtleDraw (w,h) "Tree" (tree sz)
```

In this exercise, you are to extend the above program.

3i0 (a) The following program

```
let rnd = System.Random()
let v = rnd.Next 10
```

makes a random integer between the integer $0 \leq v < 10$. Use this to make a function

`randomTree: maxStep: int -> sz: int -> turtleCmd listt`

which makes the list of turtle commands placing a tree randomly on a canvas, and the turtle to the center. Test your function by calling `turtleDraw` with such a list.

(b) Write a recursive function

`forest: maxStep: int -> sz: int -> n: int -> turtleCmd listt`

which makes `n` random trees on the canvas, and est your function by calling `turtleDraw` with such a list.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `3i.zip`
- en opgavebesvarelse i pdf-format.

Zip-filen skal indeholde:

- filen `README.txt` som er en textfil med jeres navn og dato arbejdet.
- en `src` mappe med følgende og kun følgende filer:

  3i0a.fsx og 3i0b.fsx

  svarende til afleveringsopgaverne. Funktionerne skal være dokumenteret med ifølge dokumentationsstandarden ved brug af `<summary>`, `<param>` og `<returns>` XML tagsne.
- pdf-dokumentet skal være lavet med LaTeX, benytte `opgave.tex` skabelonen, ganske kort dokumentere din løsning og indeholde 2 figurer, der viser outputgrafik fra canvas for de 2 opgaver.

God fornøjelse.