Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 4 - gruppeopgave

Jon Sporring

30. september - 8. oktober.
Afleveringsfrist: lørdag d. 8. oktober kl. 22:00.

På denne uge skal vi tage et nøjere kig på rekursion. Rekursion er et kraftigt værktøj til løsning af en række problemer, blandt andet ved behandling af lister, og kan kædes direkte sammen med induktionsbeviser til at bevise korrekthed af kode, men det kan tager tid at tilegne sig, og derfor bruger vi denne periode på at nærstudere rekursive funktioner.

Denne arbejdsseddels læringsmål er:

- Forklare forskellen på almindelig rekursion og halerekursion
- Løse et problem ved hjælp af rekursion
- Håndkøre rekursive funktioner

Opgaverne er opdelt i øve- og afleveringsopgaver. I denne periode skal I arbejde i grupper med jeres afleveringsopgaver. Regler for gruppe- og individuelle afleveringsopgaver er beskrevet i "‘Noter, links, software m.m.”’$\rightarrow$"‘Generel information om opgaver”’.

Øveopgaver (in English)

4ø0 The function `List.allPairs: 'a list -> 'b list -> ('a * 'b) list`, takes two lists and produces a list of all possible pairs. For example,

```
> List.allPairs [1..3] ['a'..'d'];;
val it: (int * char) list =
  [(1, 'a'); (1, 'b'); (1, 'c'); (1, 'd'); (2, 'a'); (2, 'b');
   (2, 'c'); (2, 'd'); (3, 'a'); (3, 'b'); (3, 'c'); (3, 'd')]
```

Make your own implementation using two `List.map` and `List.concat`.

4ø1 Write a function `length : 'a list -> int` that calculates the length of the argument list using recursion. The function should make use of pattern matching on lists.

4ø2 The following code calculates the sum $\sum_{i=0}^{n} i$ using recursion and 64-bit unsigned integeres.

```
let rec sum (n: uint64) : uint64 =
  match n with
    0UL -> 0UL
    | _ -> n+sum (n-1UL)

let n = uint64 1e4
printfn "%A" (sum n)
```

Depending on the computer, this program runs out of memory for large values of n, e.g., not many compupters can run this program when `n = uint64 1e10`. The problem is, that the algorithm is not using tail recursion.

(a) Explain why this is not tail recursion.

(b) Run the code on your computer and empirically find a small n for which the program runs out of memory on your computer.

(c) Rewrite the algorithm to become tail recursive by including the accumulated sum as the argument:

```
sum (acc: uint64) -> (n: uint64) -> uint64
```

and check that this does not run out of memory for the above identified n. Try also bigger.

(d) For which values of n would the tail recursive version break and why?

4ø3 The greatest common divisor (gcd) between two integers t and n is the largest integer c that divides both t and n with 0 as remainder. Euclid's algorithm[1] finds the greatest common divisor, using recursion, as follows:

$$\gcd(t,0) = t, \tag{1}$$
$$\gcd(t,n) = \gcd(n, t \,\%\, n), \tag{2}$$

Here % is the remainder operator (as in F#).

(a) Implement Euclid's algorithm by writing a recurive function

```
gcd : t:int -> n:int -> int
```

(b) Write down a tracing-by-hand derivation for the calls `gcd 8 2` and `gcd 2 8`.

Afleveringsopgaver (in English)

In the following you are to work with the movement of a small robot in two dimensions. The robot can be placed on integer positions `type pos = int*int`, and in each step, it can move one position up, down, left, or right.

4g0 (a) Given a source and target grid point, write the function

[1]https://en.wikipedia.org/wiki/Greatest_common_divisor

```
dist: p1: pos -> p2: pos -> int
```

which calculates the squared distance between positions p_1 and p_2. I.e., if $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ then $\text{dist}(p_1, p_2) = (x_2 - x_1)^2 + (y_2 - y_1)^2$.

(b) Given a source and a target and `dist`, write the function

```
candidates: src: pos -> src: pos -> pos list
```

which returns the list of candidate next positions, which brings the robot closer to its target. I.e., if $\text{src} = (x, y)$, then the function must consider all the neighbouring positions, $\{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$, and return those whose distance is equal to or smaller than $\text{dist}(\text{src}, \text{tg})$. This can be done with `List.filter`.

(c) Given a source and a target and by use of `candidates` the above functions, write a recursive function

```
routes: src: pos -> tg: pos -> pos list list
```

which calculates the list of all the shortest routes from `src` to `tg`. For example, the list of shortest routes from $(3,3)$ to $(1,1)$ are

```
[[(3, 3); (2, 3); (1, 3); (1, 2); (1, 1)];
 [(3, 3); (2, 3); (2, 2); (1, 2); (1, 1)];
 [(3, 3); (2, 3); (2, 2); (2, 1); (1, 1)];
 [(3, 3); (3, 2); (2, 2); (1, 2); (1, 1)];
 [(3, 3); (3, 2); (2, 2); (2, 1); (1, 1)];
 [(3, 3); (3, 2); (3, 1); (2, 1); (1, 1)]]
```

Beware, this list grows fast, the further the source and target is from each other, so you will be wise to only work with short distances. This can be done with a recursive function and a `List.map` of a `List.map`.

(d) Consider now a robot, which also can move diagonally. Extend `candidate` to also consider the diagonal positions $\{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$, and update `routes` to return the list of shortest routes only. For example, the shortest routes from $(3,4)$ to $(1,1)$ should be

```
[[(3, 4); (2, 3); (1, 2); (1, 1)];
 [(3, 4); (2, 3); (2, 2); (1, 1)];
 [(3, 4); (3, 3); (2, 2); (1, 1)]]
```

but not necessarily in that order.

(e) Optional: Make a Canvas program, which draws routes, and apply it to the routes found above.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `4g.zip`
- en opgavebesvarelse i pdf-format.

Zip-filen skal indeholde:

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

  4g0abc.fsx, 4g0d.fsx, og evt. 4g0e

  svarende til afleveringsopgaverne a-b-c, d og e. 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 figurer, der viser outputgrafik fra canvas for opgaverne.

God fornøjelse.