

Programmering og Problemløsning

Datalogisk Institut, Københavns Universitet

Arbejdsseddel 7 - gruppeopgave

Jon Sparring

21. oktober - 8. november.
Afleveringsfrist: lørdag d. 9. november kl. 23:59.

Emnerne for denne arbejdsseddel er:

- rekursion,
- pattern matching,
- sum-typer,
- endelige træer.

Opgaverne er delt 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.” → ”Generel information om opgaver”.

Øveopgaver

- 7ø0 Betragt insertion sort funktionen `isort`. Omskriv funktionen `insert` således, at den benytter sig af pattern matching på lister.
- 7ø1 Betragt bubble sort funktionen `bsort`. Omskriv den, at den benytter sig af pattern matching på lister. Funktionen kan passende benytte sig af ”nested pattern matching” i den forstand at den kan implementeres med et match case der udtrækker de to første elementer af listen samt halen efter disse to elementer.
- 7ø2 Opskriv black-box tests for de to sorteringsfunktioner og vær sikker på at grænsetilfældene dækkes (ingen elementer, et element, to elementer, samt flere elementer, sorteret, omvendt sorteret, etc.)
- 7ø3 Omskriv funktionen `merge`, som benyttes i forbindelse med funktionen `msort` (mergesort) fra forelæsningen, således at den benytter sig af pattern matching på lister.

- 7ø4 Opskriv black-box tests for sorteringsfunktionen `msort` og vær sikker på at grænsetilfældene dækkes (ingen elementer, et element, to elementer, samt flere elementer, sorteret, omvendt sorteret, etc.)
-

I det efterfølgende skal der arbejdes med `sum`-typen:

```
type weekday = Monday | Tuesday | Wednesday | Thursday
              | Friday | Saturday | Sunday
```

som repræsenterer ugens dage.

- 7ø5 Lav en funktion `dayToNumber : weekday -> int`, der givet en ugedag returnerer et tal, hvor mandag skal give tallet 1, tirsdag tallet 2 osv.
- 7ø6 Lav en funktion `nextDay : weekday -> weekday`, der givet en ugedag returnerer den næste dag, så mandag skal give tirsdag, tirsdag skal give onsdag, osv, og søndag skal give mandag.
- 7ø7 Lav en funktion `numberToDay : n : int -> weekday option`, sådan at `numberToDay n` returnerer `None`, hvis `n` ikke ligger i intervallet `1..7`, og ellers returnerer ugedagen `Some d`. Det skal gælde, at `numberToDay (dayToNumber d) ==> Some d` for alle ugedage `d`.
-

- 7ø8 Ved at benytte biblioteket `ImgUtil`, som beskrevet i forelæsningen, er det muligt at tegne simpel liniegrafik samt fraktaler, som f.eks. Sierpinski-fraktalen, der kan tegnes ved at tegne små firkanter bestemt af et rekursivt mønster. Koden for Sierpinski-trekanten er givet som følger:

```
open ImgUtil

let rec triangle bmp len (x,y) =
  if len < 25 then setBox blue (x,y) (x+len,y+len) bmp
  else let half = len / 2
        do triangle bmp half (x+half/2,y)
        do triangle bmp half (x,y+half)
        do triangle bmp half (x+half,y+half)

do runSimpleApp "Sierpinski" 600 600 (fun bmp -> triangle bmp
512 (30,30) |> ignore)
```

Tilpas funktionen således at trekanten tegnes med røde streger samt således at den kun tegnes 2 rekursionsniveauer ned. Hint: dette kan gøres ved at ændre betingelsen `len < 25`.

- 7ø9 I stedet for at benytte `ImgUtil.runSimpleApp` funktionen skal du nu benytte `ImgUtil.runApp`, som giver mulighed for at din løsning kan styres ved brug af tastaturet. Funktionen `ImgUtil` har følgende type:

```
val runApp : string -> int -> int
            -> (int -> int -> 's -> System.Drawing.Bitmap)
            -> ('s -> System.Windows.Forms.KeyEventArgs
                -> 's option)
            -> 's -> unit
```

De tre første argumenter til `runApp` er vinduets titel (en streng) samt vinduets initiale vidde og højde. Funktionen `runApp` er parametrisk over en brugerdefineret type af tilstande ('s). Antag at funktionen kaldes som følger:

```
runApp title width height draw react init
```

Dette kald vil starte en GUI applikation med titlen `title`, vidden `width` og højden `height`. Funktionen `draw`, som brugeren giver som 4. argument kaldes initielt når applikationen starter og hver gang vinduets størrelse justeres eller ved at funktionen `react` er blevet kaldt efter en tast er trykket ned på tastaturet. Funktionen `draw` modtager også (udover værdier for den aktuelle vidde og højde) en værdi for den brugerdefinerede tilstand, som initielt er sat til værdien `init`. Funktionen skal returnere et bitmap, som for eksempel kan konstrueres med funktionen `ImgUtil.mk` og ændres med andre funktioner i `ImgUtil` (f.eks. `setPixel`).

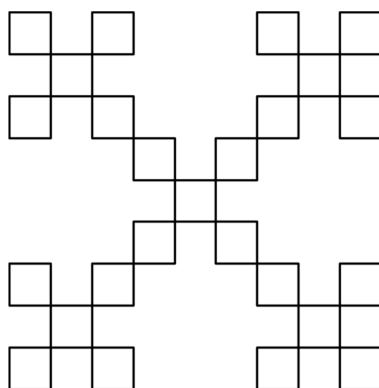
Funktionen `react`, som brugeren giver som 5. argument kaldes hver gang brugeren trykker på en tast. Funktionen tager som argument:

- en værdi svarende til den nuværende tilstand for applikationen, og
- et argument der kan benyttes til at afgøre hvilken tast der blev trykket på.¹

Funktionen kan nu (eventuelt) ændre på dens tilstand ved at returnere en ændret værdi for denne.

Tilpas applikationen således at dybden af fraktalen kan styres ved brug af piletasterne, repræsenteret ved værdierne `System.Windows.Forms.Keys.Up` og `System.Windows.Forms.Keys.Down`.

7ø10 Med udgangspunkt i øvelsesopgave 7ø8 skal du i denne opgave implementere en GUI-applikation der kan tegne en version af X-fraktalen som illustreret nedenfor (eventuelt i en dybde større end 2).



Bemærk at det ikke er et krav, at dybden på fraktalen skal kunne styres med piletasterne, som det er tilfældet med Sierpinski-fraktalen i øvelsesopgave 7ø9.

Afleveringsopgaver

H.C. Andersen (1805-1875) is a Danish author who wrote plays, travelogues, novels, poems, but perhaps is best known for his fairy tales. An example is Little Claus and Big Claus (Danish: Lille

¹Hvis `e` har typen `System.Windows.Forms.KeyEventArgs` kan betingelsen `e.KeyCode = System.Windows.Forms.Keys.Up` benyttes til at afgøre om det var tasten "Up" der blev trykket på.

Claus og store Claus), which is a tale about a poor farmer, who outsmarts a rich farmer. A translation can be found here: http://andersen.sdu.dk/vaerk/hersholt/LittleClausAndBigClaus_e.html. It starts like this:

Hans Christian Andersen's "Lille Claus og Store Claus" translated by Jean Hersholt.

In a village there lived two men who had the self-same name. Both were named Claus. But one of them owned four horses, and the other owned only one horse; so to distinguish between them people called the man who had four horses Big Claus, and the man who had only one horse Little Claus. Now I'll tell you what happened to these two, for this is a true story."

In this assignment, you are to work with simple text processing, analyse the statistics of the text, and use this to generate a new text with similar statistics.

7g0 The script `readFile.fsx` reads the content of the text file `readFile.fsx`. Convert this script into a function which can read the content of any text file and has the following type:

```
readText : filename:string -> string
```

7g1 Write a function that converts a string, such that all letters are converted to lower case, and removes all characters except a...z. It should have the following type:

```
convertText : src:string -> string
```

7g2 Write a function,

```
histogram : src:string -> int list
```

which counts occurrences of each lower-case letter of the English alphabet in a string and returns a list. The first element of the list should be the count of 'a's, second the count of 'b's etc.

7g3 The script `sampleAssignment.fsx` contains the function

```
randomString : hist:int list -> len:int -> string
```

which generates a string of a given length, and contains random characters distributed according to a given histogram. Modify the code to use your histogram function from Exercise 7g2.

Further, write a program, which reads the text `littleClausAndBigClaus.txt` using `readText`, converts it using `convertText`, and calculates its histogram and generates a new random string using `histogram` and `randomString`. Test the quality of your code by comparing the histograms of the two texts.

7g4 Write a function

```
cooccurrence : src:string -> int list list
```

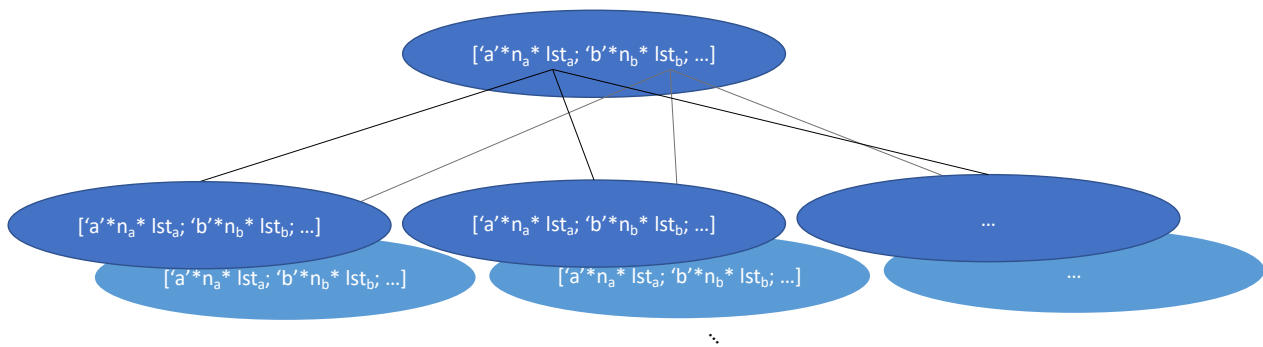


Figure 1: An illustration of a list of values of the type Tree.

which counts occurrences of each pairs of lower-case letter of the English alphabet including space in a string and returns a list of lists (a table). In the return list, the first element should be a list of the counts of 'a' being the initial character, i.e., how many times “aa”, “ab”, “ac”,... was observed. The second list should contain the counts of combinations starting with 'b', i.e., how many times “ba”, “bb”, “bc”,... was observed and so on. The function should include overlapping pairs, for example, the input string “abcd” has the pairs “ab”, “bc”, and “cd”.

7g5 Write a function

```
fstOrderMarkovModel : cooc:int list list -> len:int -> string
```

which generates a random string of length len, whose character pairs are distributed according to a user specified cooccurrence histogram cooc.

Use the function developed in Exercise 7g1 and 7g4, and test your function by generating a random string, whose character pairs are distributed as the converted characters in H.C. Andersen’s fairy tale, “Little Claus and Big Claus”. Calculate the cooccurrence histogram for the random string, and compare this with the original cooccurrence histogram.

7g6 Write a function that counts occurrences of each word in a string and returns a list. The counts must be organized as a list of trees using the following Tree type:

```
type Tree = Node of char * int * Tree list
```

An illustration of a value of this type is shown in Figure 1 Words are to be represented as the sequence of characters from the root til a node. The associated integer to each node counts the occurrence of a word ending in that node. Thus, if the count is 0, then no words with that end point has occurred. For example, a string with the words “a abc ba” should result in the following tree,

```
[Node ('a', 1, [Node ('b', 0, [Node ('c', 1, [])])]);
 Node ('b', 0, [Node ('a', 1, [])])]
```

Notice, the counts are zero for the combinations “ab” and “b”, which are words not observed in the string. The function must have the type:

```
wordHistogram : src:string -> Tree list
```

Write a program which reads the text `littleClausAndBigClaus.txt`, discard all characters that are not in `['a'..'z', 'A'..'Z', ' ']`, convert all the remaining characters to lower case, and calculate the occurrence of the remaining words as a `Tree list` type.

7g7 For a given value of a `Tree` type, see Exercise 7g6, write a function

```
randomWords : wHist:Tree list -> nWords:int -> string
```

which generates a string with `nWords` number of words randomly selected to match the word distribution in `wHist`.

Use the function developed in Exercise 7g1 and 7g6, and test your function by generating a random string, whose words are distributed as the converted characters in H.C. Andersen's fairy tale, "Little Claus and Big Claus". Calculate the word histogram for the random text, and compare this with `wHist`.

7g8 Write a short report, which

- is no larger than 5 pages;
- contains a brief discussion on how your implementation works, and if there are any possible alternative implementations, and in that case, why you chose the one, you did;
- includes output that demonstrates that your solutions work as intended.

Afleveringen skal bestå af

- en zip-fil, der hedder `7g_<navn>.zip` (f.eks. `7g_jon.zip`)
- en pdf-fil, der hedder `7g_<navn>.pdf` (f.eks. `7g_jon.pdf`)

Zip-filen `7g_<navn>.zip` skal indeholde en og kun en mappe `7g_<navn>`. I den mappe skal der ligge en `src` mappe og filen `README.txt`. I `src` skal der ligge følgende og kun følgende filer: `7g0.fsx`, `7g1.fsx` og `7g2.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. Filen `README.txt` skal ganske kort beskrive, hvordan koden oversættes og køres. Pdf-filen skal indeholde jeres rapport oversat fra \LaTeX . Husk at pdf-filen skal uploades ved siden af zip-filen på Absalon.

God fornøjelse.