Getting Started

Jon Sporring

July 4, 2019

1 Introduktion

2 Opgave(r)

2.1 Check installation, get accustomed to interactive and compile mode

1.1. Start en interaktiv F# session og indtast følgende (efterfulgt af ny linie):

Listing 1: My first F#.

```
3.14+2.78;;
```

Beskriv (for dig selv), hvad F# gjorde, og hvis der opstod en fejl, find fejlen og gentag.

1.2. Gentag øvelsen ovenfor, men denne gang indtast udtrykket i Emacs og gem det i en fil med suffix .fsx. Kør filen med fsharpi og fsharpc+mono. Overvej om resultatet er som forventet, og hvis ikke, forklar hvorfor.

1.3. Prøv følgende udtryk i F#,

Listing 2: Problematic F#.

```
3 + 1.0;;
```

og forklar resultatet. Forbedr evt. udtrykket.

1.4. Betragt F#-udtrykket 164uy+230uy. Forklar hvad "uy" betyder, udregn udtrykket i F# og diskutér resultatet.

1.5. Opskriv et F#-udtryk, som indicerer det 3. element og substrengen fra det 2. til 4. elemente i strengen “abcdef”.

1.6. Opskriv to F#-udtryk, som ved brug af indiceringssyntaksen udtrækker 1. og 2. ord i strengen “hello world”.

2.2 Get accustomed to F# syntax and precedence

2.1. Indtast følgende program i en tekstfil, og oversæt og kør programmet

Listing 3: Værdibindinger.

```
let a = 3
let b = 4
let x = 5
printfn "%A * %A + %A = %A" a x b (a * x + b)
```

Forklar hvad parentesen i kaldet af printfn funktionen gør godt for. Tilføj en linje i programmet, som udregner udtrykket $ax+b$ og binder resultatet til y, og modificer kaldet til printfn så det benytter denne nye binding. Er det stadig nødvendigt at bruge parentes?

2.2. Listing 3 benytter F#'s letvægtssyntaks (Lightweight syntax). Omskriv programmet (enten med eller uden y bindingen), så det benytter regulær syntaks.

2.3. Følgende program,

Listing 4: Strenge.

```
let firstName = "Jon"
let lastName = "Sporring" in let name = firstName + " " +
    lastName;;
printfn "Hello %A!" name;;
```

skulle skrive "Hello Jon Sporring!" ud på skærmen, men det indeholder desværre fejl og vil ikke oversætte. Ret fejlen(e). Omskriv programmet til en linje (uden brug af semikolonner). Overvej hvor mange forskellige måder, dette program kan skrives på, hvor det stadig gør brug af bindingerne firstName lastName name og printfn funktionen.

2.4. Tilføj en funktion

```
f : a:int -> b:int -> x:int -> int
```

til Listing 3, hvor a, b og x er argumenter til udtrykket $ax+b$, og modificer kaldet til printfn så det benytter funktionen istedet for udtrykket (a * x + b).

2.5. Brug funktionen udviklet i Opgave 2.4, således at du udskriver værdien af funktionen for $a=3$, $b=4$ og $x=0\ldots5$ ved brug af 6 printfn kommandoer. Modificer nu dette program vha. af en for løkke og kun en printfn kommando. Gentag omskrivningen men nu med en while løkke.

2.6. Lav et program, som udskriver 10-tabellen på skærmen, således at der er 10 søjler og 10 rækker formateret som

	1	2	…	10
1	1	2	…	10
2	2	4	…	20
⋮				
10	10	20	…	100

hvor venstre søjle og første række angiver de tal som er ganget sammen. Du skal benytte to for løkker, og feltbredden for alle tallene skal være den samme.

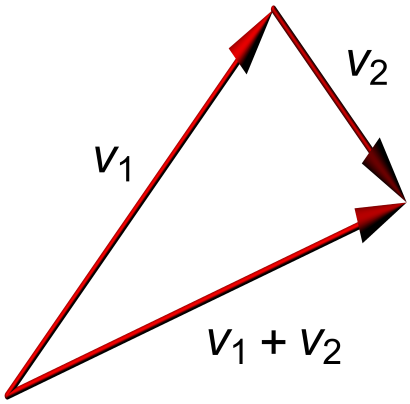

Figure 1: Illustration of vector addition in two dimensions.

2.3 Advanced exercise using modules

Opgaven omhandler to-dimensionelle vektorer. En to-dimensionel vektor (herefter blot vektor) er et geometrisk objekt som består af en retning og en længde. Typisk repræsenteres vektorer som par af tal $\vec{v} = (x, y)$, hvor længden og retning findes ved,

$$\text{len}(\vec{v}) = \sqrt{x^2 + y^2} \tag{1}$$
$$\text{ang}(\vec{v}) = \text{atan2}(y, x) \tag{2}$$

Vektorens ender kaldes hhv. hale og spids, og når halen placeres i $(0,0)$, så vil spidsen have koordinat (x, y). Vektorer har en række standardoperatorer,

$$\vec{v}_1 = (x_1, y_1) \tag{3}$$
$$\vec{v}_2 = (x_2, y_2) \tag{4}$$
$$a\vec{v}_1 = (ax_1, ay_1) \tag{5}$$
$$\vec{v}_1 + \vec{v}_2 = (x_1 + x_2, y_1 + y_2) \tag{6}$$
$$\vec{v}_1 \cdot \vec{v}_2 = x_1 x_2 + y_1 y_2 \tag{7}$$

Addition kan tegnes som vist i Figure 1.

3.1. Skriv et bibliotek vec2d.fs, som implementerer signatur filen givet i Listing 5.

Listing 5 vec2d.fsi:
A signature file.

```
/// A 2 dimensional vector library.
/// Vectors are represented as pairs of floats
module vec2d
/// The length of a vector
val len : float * float -> float
/// The angle of a vector
val ang : float * float -> float
/// Multiplication of a float and a vector
val scale : float -> float * float -> float * float
/// Addition of two vectors
val add : float * float -> float * float -> float * float
/// Dot product of two vectors
val dot : float * float -> float * float -> float
```

3.2. Skriv en White-box afprøvning af biblioteket.

3.3. Punkter på en cirkel med radius 1 kan beregnes som $(\cos\theta, \sin\theta)$, $\theta \in [0, 2\pi)$. Betragt det lukkede polygon, som består af $n > 1$ punkter på en cirkel, hvor $\theta_i = \frac{2\pi i}{n}$, $i = 0..(n-1)$.

(3.3.1) Skriv et program med en funktion,

polyLen : n:int -> float

som benytter ovenstående bibliotek til at udregne længden af polygonet. Længden udregnes som summen af længden af vektorerne mellem nabopunkter. Programmet skal desuden udskrive en tabel af længder for et stigende antal værdier n, og resultaterne skal sammenlignes med omkredsen af cirklen med radius 1.

(3.3.2) Udform en hypotese ud fra tabellen for længden af polygonet når $n \to \infty$.

3.4. Biblioteket vec2d tager udgangspunkt i en repræsentation af vektorer som par (2-tupler). Lav et udkast til en signaturfil for en variant af biblioteket, som ungår tupler helt. Diskutér eventuelle udfordringer og større ændringer, som varianten ville kræve både for implementationen og programmet.

2.4 Recursion in F#

4.1. Skriv en funktion, fac : n:int -> int, som udregner fakultetsfunktionen $n! = \prod_{i=1}^{n} i$ vha. rekursion.

4.2. Skriv en funktion, sum : n:int -> int, som udregner summen $\sum_{i=1}^{n} i$ vha. rekursion. Lav en tabel som i Opgave 3i.0 og sammenlign denne implementation af sum med while-implementation og simpleSum.

4.3. Den største fællesnævner mellem 2 heltal, t og n, er det største heltal c, som går op i både t og

n med 0 til rest. Euclids algoritme[1] finder den største fællesnævner vha. rekursion:

$$\gcd(t,0) = t, \tag{8}$$
$$\gcd(t,n) = \gcd(n, t \ \% \ n), \tag{9}$$

hvor % er rest operatoreren (som i F#).

(4.3.1) Implementer Euclids algoritm, som den rekursive funktion

```
gcd : int -> int -> int
```

(4.3.2) lav en white- og black-box test af den implementerede algoritme,

(4.3.3) Lav en håndkøring af algoritmen for `gcd 8 2` og `gcd 2 8`.

4.4. Lav dine egen implementering af `List.fold` og `List.foldback` ved brug af rekursion.

2.5 Lists and patterns

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

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

5.3. Brug `List.map` og `reverseApply` (fra Opgave 5.2) til at lave en funktion `applylist : ('a -> 'b) list -> 'a -> 'b list`, der anvender en liste af funktioner på samme element for at returnere en liste af resultater.

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

5.5. 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]`.

2.6 Individual hand-in assigment: Random Text

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

> "LITTLE CLAUS AND BIG CLAUS a translation of hans christian andersen's 'lille claus og store claus' 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

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

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

6.1. character histograms of a HCAndersen text, character transition, and random text generation using a 1st order Markov model. It should be tested using a simple test set and with a complicated test set by recalculating the histograms and transitions for the generated text and comparing with the original.

(6.1.1) Write a program, that reads an UTF-8 text from a file as a string, and which has the following type:

readText : filename:string -> string

(6.1.2) Write a program that converts a string, such that all letters are converted to lower case, and which removes all characters except a...z. Is should have the following type:

convertText : src:string -> string

(6.1.3) Write a program that counts occurences 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. The function must have the type:

histogram : src:string -> character list

(6.1.4) Write a program that counts occurences of each pairs of lower-case letter of the English alphabet in a string and returns a list of lists. The first list should be the count of 'a' followed by 'a's, 'b's, etc., second list should be the count of 'b' followed by 'a's, 'b's, etc. etc.

cooccurence : src:string -> character list list

Der skal også laves:

- En rapport (maks 2 sider)
- Unit-tests
- Implementation skal kommenteres jævnfør kommentarstandarden for F#