Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 2 - gruppeopgave

Jon Sporring

16. - 24. september.
Afleveringsfrist: lørdag d. 24. september kl. 22:00.

Med denne arbejdsseddel gælder for 1 uge, og med den skifter vi perspektiv til funktionsprogrammering. Derfor vil vi undgå mutérbare værdier og bruge rekursion til løkker. Curriculum for opgaverne er [Sporring, kapitel 3-6]. Læringsmålene for denne uge er: Denne arbejdsseddels læringsmål er:

- at kunne strukturere kode vha. funktioner og sum typer,
- at kunne håndkøre simple programmer.
- at kunne dokumentere kode vha. XML standarden.

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."'→"'Generel information om opgaver"'.

Øveopgaver (in English)

2ø0 The following program,

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

is supposed to write "Hello Jon Sporring!" to the screen, but unfortunately, it contains at least one mistake. Correct the mistake(s) and rerun the program.

2ø1 Perform a trace-by-hand of the following program

```
let a = 3.0
let b = 4.0
let f x = a * x + b

let x = 2.0
let y = f x
printfn "%A * %A + %A = %A" a 2.0 b y
```

2ø2 Consider the factorial-function,

$$n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot \ldots \cdot n \tag{1}$$

(a) Write a function

```
fac : n:int -> int
```

which uses recursion to calculate the factorial-function as (1).

(b) Write a program, which asks the user to enter the number `n` using the keyboard, and which writes the result of `fac n`.

(c) Make a new version,

```
fac64 : n:int64 -> int64
```

which uses `int64` instead of `int` to calculate the factorial-function. What are the largest values n, for which `fac` and `fac64` respectively can correctly calculate the factorial-function for?

2ø3 Using Steps 1, 3, 5, 7, and 8 from the 8-step guide,

(a) write a recursive function which takes two integer arguments x and n and returns the value x^n.

(b) write another function wich takes one argument (x,n) and calls the former.

Document both functions using the `<summary>`, `<param>`, and `<returns>` XML tags. Consider what should happen, if $n < 0$, and whether there is any significant difference between the call of the two functions.

I the following, you are to work with the discrimative union:

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

which represents the days of the week.

2ø4 Make a function `dayToNumber : weekday -> int` which given a `weekday` returns an integer, sucht that Monday is 1, Tuesday 2, etc.

2ø5 Make a function `nextDay : weekday -> weekday` which given a day, returns the next day, i.e., Tuesday is the next day of Monday, and Monday is the next day of Sunday.

2ø6 Make a function `numberToDay : n : int -> weekday option` which given an integer $1 \ldots 7$ returns the weekday Monday... Sunday.

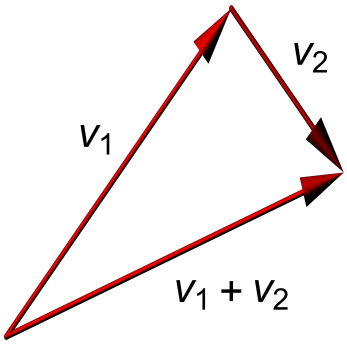

Figure 1: Illustration of vector addition in two dimensions.

Afleveringsopgaver (in English)

This assignment is about 2-dimensional vectors. A 2-dimensional vector or just a vector is a goemetrical object consisting of a direction and a length. Typically, vectors are represented as a coordinate par $\vec{v} = (x, y)$, where the length and direction is found as

$$\text{len}(\vec{v}) = \sqrt{x^2 + y^2} \tag{2}$$
$$\text{ang}(\vec{v}) = \text{atan2}(y, x) \tag{3}$$

For technical reasons, we here use atan2 instead of the usual atan function. The atan2 function is found in F# as `System.Math.Atan2`, which takes the (y,x) tuple argument. The vector's ends are called its tail and tip, and when the tail is placed in $(0,0)$, then its tip will be in the (x,y). Vectors have a number of standard operations on them:

$$\vec{v}_1 = (x_1, y_1) \tag{4}$$
$$\vec{v}_2 = (x_2, y_2) \tag{5}$$
$$a\vec{v}_1 = (ax_1, ay_1) \tag{6}$$
$$\vec{v}_1 + \vec{v}_2 = (x_1 + x_2, y_1 + y_2) \tag{7}$$
$$\vec{v}_1 \cdot \vec{v}_2 = x_1 x_2 + y_1 y_2 \tag{8}$$

Addition can be drawn as shown in Figure 1. Rotation of a vector by the a counter clockwise and around its tail can be done as,

$$R_a \vec{v}_1 = (x\cos(a) - y\sin(a), x\sin(a) + y\cos(a)) \tag{9}$$

The trigonometric functions are found as `System.Math.Cos` and `System.Math.Sin`, and they both take an angle in radians as the argument. The constant π is found in `System.Math.PI`.

2g0 Using Steps 1, 3, 5, 7, and 8 from the 8-step guide to write a small set of functions in F#:

(a) addition of vectors

```
add: float * float -> float * float -> float * float
```

(b) multiplication of a vector and a constant

```
mul: float * float -> float -> float * float
```

(c) dot-product of two vectors

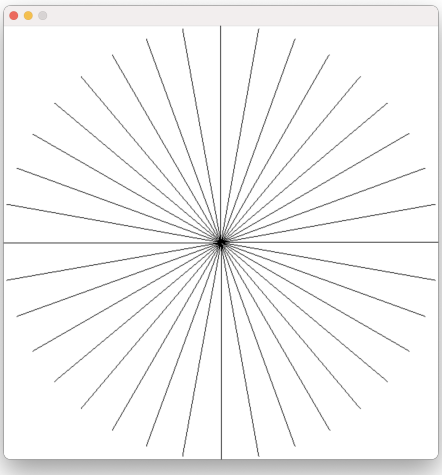

Figure 2: 36 radial lines from the center of a canvas.

```
dot: float * float -> float * float -> float
```

(d) rotation of a vector

```
rot: float * float -> float -> float * float
```

The functions are to be documented using the `<summary>`, `<param>`, and `<returns>` XML tags.

2g1 Using Canvas, you are to draw vectors.

(a) make a function

```
toInt: float * float -> int * int
```

which takes a vector of floats and returns a vector of ints.

(b) Using `add` and `toInt`, make a function

```
setVector: canvas -> color -> int * int -> int * int -> unit
```

which takes a canvas, a color, a vector, and a position for its tail and draws it as a line with `setLine` on the canvas. Demonstrate that this works by drawing the canvas with `show`.

(c) make a function using `rot` and `seVector`

```
draw: int -> int -> canvas
```

which creates a canvas with a given width and height, adds 36 spokes as illustrated in Figure 2, and returns the canvas. Demonstrate that this works by drawing the canvas with `show`.

(d) Optional: Use these in `runApp` to make an interactively rotating set of spokes as follows: Extend `draw` with a float state parameter `s`, which draws the spokes with the angular offset `s`. Add a reaction function `react` which changes the offset by ± 0.01 when the right and left arrow key are pressed respectively.

The functions are to be documented using the `<summary>`, `<param>`, and `<returns>` XML tags.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `2g.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:

  2g0.fsx og 2g1.fsx

  svarende til afleveringsopgaverne
- en `tex` mappe med følgende og kun følgende filer:

  2g.tex og og to screenshots et fra hver af 2g1b.fsx og 2g1c.fsx i png-format.

LaTeXdokumentet 2g.tex skal benytte `opgave.tex` skabelonen og ganske kort dokumentere jeres løsning. Et Screenshot af Canvas vinduet skal inkluderes i dokumentet.

Afleveringen skal bestå af

- en zip-fil, der hedder `2g_<navn>.zip` (f.eks. `2g_jon.zip`)

Zip-filen `2g_<navn>.zip` skal indeholde en `src` mappe, filen `README.txt` *og filen "group.txt", der indeholder jeres kuid'er, ét per linje*. I `src` skal der ligge følgende og kun følgende filer: `2g0.fsx` og `2g1.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. Udover selve koden skal besvarelser indtastes som kommentarer i de fsx-filer, de hører til. Filen `README.txt` skal ganske kort beskrive, hvordan koden oversættes og køres, og eventuelle Black-box, White-box og håndkøringsresultater når relevant.

God fornøjelse.