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

Jon Sporring

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

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]. 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.

In the following, you are to work with the discriminated union `weekday`:

```
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, such that Monday is 1, Tuesday is 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 in the range 1...7 returns one of the weekdays Monday...Sunday as an option type. An integer not in the range, i.e. < 1 or > 7 should return `None`.
Examples:
The call `numberToDay 1` should return `Some Monday` and The call `numberToDay 42` should return `None`.

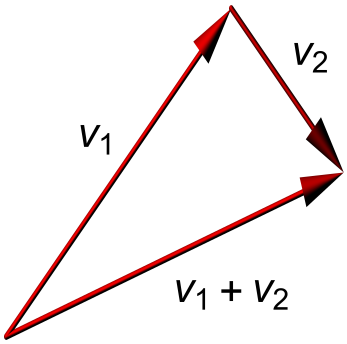

Figure 1: Illustration of vector addition in two dimensions.

2ø7 In the following, you are to work with tuples.

(a) Make a type abbreviation called `vec3`, which is a 3-tuple of `floats`.

(b) Make a value of `vec3`.

(c) Make a function, which takes a `vec3` as argument and returns the squared sum of its elements, and test it on the value, you created. Consider the different ways, the function's type could be written, and what their qualitative differences would be.

Afleveringsopgaver (in English)

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

$$\vec{v}_1 = (x_1, y_1) \tag{2}$$
$$\vec{v}_2 = (x_2, y_2) \tag{3}$$
$$\vec{v}_1 + \vec{v}_2 = (x_1 + x_2, y_1 + y_2) \tag{4}$$
$$a\vec{v}_1 = (ax_1, ay_1) \tag{5}$$

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

$$R_a\vec{v}_1 = (x\cos(a) - y\sin(a), x\sin(a) + y\cos(a)) \tag{6}$$

In F#, the trigonometric functions are found in `cos` and `sin`, and they both take an angle in radians as the argument. The constant π is found in `System.Math.PI`. In the following, we will use the type abbreviation:

```
type vec = float * float
```

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

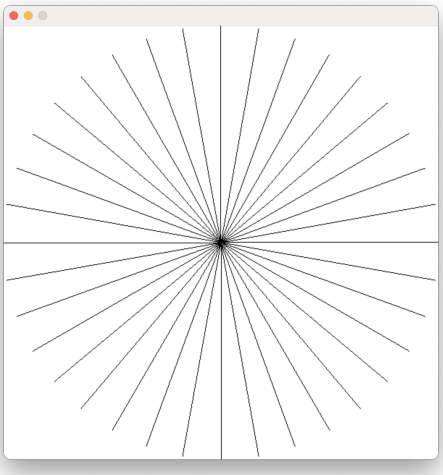

Figure 2: 36 radial lines from the center of a canvas.

(a) addition of vectors (4)

```
add: vec -> vec -> vec
```

(b) multiplication of a vector and a constant (5)

```
mul: vec -> float -> vec
```

(c) rotation of a vector (6)

```
rot: vec -> float -> vec
```

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

2g1 Using Canvas, you are to draw vectors. For this,

(a) Make a function

```
toInt: vec -> 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 -> vec -> vec -> unit
```

which takes a canvas, a color, a vector `v`, and a position `p` and draws a line from `p` to `p+v` using `setLine`. Demonstrate that this works by creating a horizontal vector with its tail at the center of the canvas, and show it on screen using `show`.

(c) Using `rot` and `setVector` make a function

```
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 showing the canvas on screen 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 screenshots der viser resultaterne fra hhv. 2g1b.fsx, 2g1c.fsx og evt. 2g1d.fsx i png-format.

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

God fornøjelse.