Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Arbejdsseddel 1 - individuel opgave

Jon Sporring

2. september - 16. september.
Afleveringsfrist: lørdag d. 17. september kl. 22:00.

Velkommen til kurset "Programmering og Problemløsning"

Kurset består af forelæsninger og øvelser. Forelæsningerne er både video-, on-campus og streamede forelæsninger og gives af forelæserne Jon Sporring (kursuskoordinator), Ken Friis Larsen og Fritz Henglein. De vil omhandle de væsentligste teoretiske elementer i programmering for begyndere. Til øvelserne er alle fordelt på øvelseshold og fokuserer på de praktiske elementer i programmering for begyndere. Til hvert øvelseshold er der en instruktor, og øvelsesholdene vil være on-campus. Kurset ligger i skemagruppe A (tirsdag formiddage og torsdag).

Dette er kursets første *arbejdsseddel*. Vi har 16 undervisningsuger, og de fleste arbejdssedler vil svare en undervisningsuge. Arbejdssedlerne vil beskrive pensum, som gennemgåes til forelæsningerne, øvelsesopgaver, som der vil blive arbejdet med under øvelserne, og afleveringsopgaver, som der også bliver tid til at kigge på til øvelserne. Pensum angives under "Forelæsnings- og læseplan" på Absalon og evt. tilhørende materiale finder under Absalonpunktet "Noter, links, software m.m.". Ca. halvdelen af afleveringsopgaverne bliver individuelle opgaver, alle andre opgaver løses i grupper.

Bemærk at langt de fleste opgaver på disse arbejdsedler vil være på engelsk, da vi er igang med at oversætte dem til Fsharp bogen. Denne omgivende tekst har vi dog valgt at beholde på dansk.

Denne arbejdsseddels læringsmål er:

- Lave simple programmer i dotnet/F#,
- skrive en rapport i LaTeX vha. Overleaf,
- aflevere en opgave via Absalon.

Opgaverne er opdelt i øve- og afleveringsopgaver. I denne periode skal I arbejde individuelt med jeres afleveringsopgaver. Regler for gruppe- og individuelle afleveringsopgaver er beskrevet i "'Noter, links, software m.m."'→"'Generel information om opgaver"'.

Øveopgaver (in English)

1ø0 Start the command line (or terminal on MacOS). Use the `cd`-command to navigate to a suitable directory for your work. (e.g. the Documents folder). Use the

```
mkdir <name>
```

command to create a new directory from the command line. Replace `<name>` with the name of your new directory. Use `ls` or `dir` to verify that the new directory is empty. Locate the same directory with the Graphical User Interface.

1ø1 Start an interactive F# session on the command line by writing

```
dotnet fsi
```

and type the following ending with a newline:

```
3+2;;
```

Describe what F# did and if there was an error, find it and repeat.

1ø2 Start `dotnet fsi` and interactive mode, write a function `subInt a b`, where the arguments are integers, and which returns $a - b$. Write a similar function `subFloat a b` where the arguments are floats. In the type-signature returned by F#, how can you see that these functions have 2 arguments and what their types are?

1ø3 Start an editor and write a program which prints "hello world" on the screen. Save the file as a `.fsx` file. Execute the program from the command line using

```
dotnet fsi <filename>
```

where `<filename>` is the name of the file you chose, and verify that it prints as expected.

1ø4 Write a program in an editor, which

(a) Writes `"Hello, what is your name:"` to the screen

(b) Reads the users name from the keyboard

(c) Prints `"Hello <name>"` to the screen where `<name>` is replaced by what the user enters.

Save the file, and run the program using `"dotnet fsi <filename>"` and verify that it does as you expect.

1ø5 Type the following expression in F# interactive mode, `"123" + "456";;` including the quotation marks and explain the result.

1ø6 Use a `while`-loop and a mutable value to write a program, which prints the numbers 1…10 on the screen.

1ø7 Use a recursive-loop and pattern recognition to write a program, which prints the numbers 1…10 on the screen.

1ø8 Create an empty canvas of width 400 pixels and height 600 pixels (400×600). Draw a red, green, blue, and yellow line on the left, bottom, right, and top edge, and a black 200×300 rectangle in the middle.

1ø9 Consider points on a circle is give by the coordinate functions

$$x(t) = x_0 + r\cos(t) \tag{1}$$
$$y(t) = y_0 + r\sin(t) \tag{2}$$

whose center is (x_0, y_0), r is its radius, and where $0 \leq t < 2\pi$. Use these equations to make a canvas of size 400×400 with an approximation of a circle centered at $(200, 200)$ and with a radius of 100. The approximation should consists of straight lines connecting $(x(t_i), y(t_i))$ with $(x(t_{i+1}), y(t_{i+1}))$ for $t = [0, 0.1, 0.2, \ldots, 2\pi)$.

1ø10 Make a program, which draws a 20×20 square in the center of a canvas of size 400×400. When the user presses space, or an arrow key, the program should write to the terminal, which key has been pressed.

1ø11 Extend the program in Exercise 1ø10, such that when the arrow keys are pressed, then the square is moved in the direction of the arrow pressed.

1ø12 Make a new project in `https://overleaf.com`, and compile the default document. Download its pdf, and download the project. Verify that the pdf and the `.tex` file that you have downloaded, looks like what you have entered in overleaf.

1ø13 Make a new project in `https://overleaf.com`, and update the default `.tex`-file to contain as little as possible, while still being able to compile. What is the shortest LaTeXprogram possible?

1ø14 Make a new project in `https://overleaf.com`, and write a short document in LaTeX. The report should as minimum contain:

- A title produced using `\maketitle`,
- A section with a section title using `\section`,
- One or more figures of images from your program, using the `figure`-environment. All figures must include a caption text using `\caption`.
- A reference to the figure using the `\label`–`\ref` pair.
- The Danish letters 'æ', 'ø', and 'å'.

1ø15 Make a new project in `https://overleaf.com`, replace the relevant file(s) with `opgave.tex` from Absalon. Compile it and check that the is correct.

Afleveringsopgaver (in English)

1i0 Modify the following program (`quickStartRecursiveInput.fsx`):

```
let rec readNonZeroValue () =
  let a = int (System.Console.ReadLine ())
  match a with
    0 ->
      printfn "Error: zero value entered. Try again"
      readNonZeroValue ()
    | _ ->
      a
printfn "Please enter a non-zero value"
let b = readNonZeroValue ()
printfn "You typed: %A" b
```

such that instead of asking the user for a non-zero value, repatedly asks the user for the name of a programming language. If the user enters `"quit"`, then the program should stop. If the user enters `"fsharp"`, then the program should write `"fsharp is cool"`. In all other cases, program should write `"I don't known <name>"`, where `"<name>"` is the string, the user entered. Example of a user dialogue is:

```
% dotnet fsi fsharpIsCool.fsx
Please enter the name of a programming language:
c
I don't know "c"
Please enter the name of a programming language:
fsharp
Fsharp is cool
Please enter the name of a programming language:
quit
```

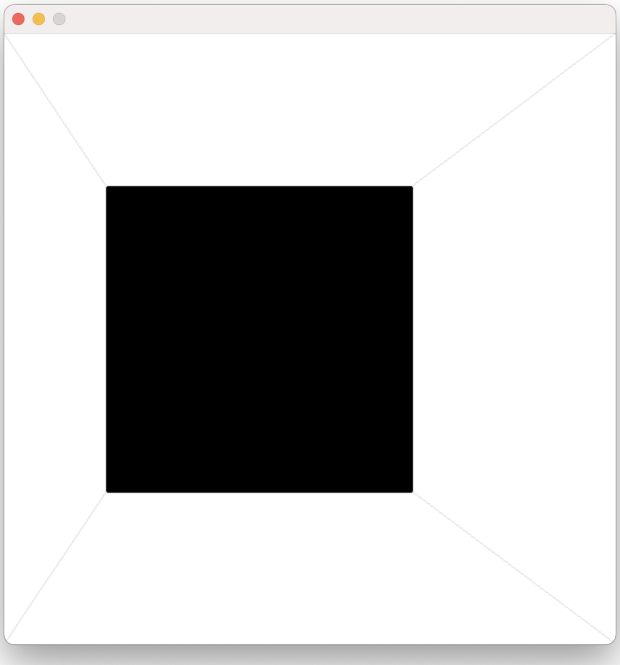
1i1 Make a simple, interactive program using the Canvas library, and which could be the start of a game. You are to modify the demo-program `color_boxes.fsx`, such that instead of colored boxes the resulting program:

(a) Creates an interactive canvas

(b) Draws a black box on a white background with grey lines from the corners of the box to the nearest corner of the canvas-

(c) When the user presses the right or left arror key, then the box with lines are moved to the right and left respectively.

An example of how this could look is:



The functions `runApp`, `setLine`, and `getKey` are most likely essential for this task.

Krav til afleveringen

Afleveringen skal bestå af

- en zip-fil, der hedder `1i.zip`
- en rapport i pdf-format.

Zip-filen skal indeholde:

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

  1i0.fsx og 1i1.fsx

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

  1i.tex og og et screenshot af jeres applikation i 1i1.fsx i png-format.

LaTeXdokumentet 1i.tex skal benytte `opgave.tex` skabelonen og ganske kort dokumentere jeres løsning. Et Screenshot af Canvas vinduet skal inkluderes i dokumentet.

God fornøjelse.