

Programmering og Problemløsning

Datalogisk Institut, Københavns Universitet

Arbejdsseddel 3 - individuel opgave

Jon Sparring

20. - 26. september.
Afleveringsfrist: onsdag d. 26. september kl. 22:00

I denne periode skal I arbejde individuelt. Regler for individuelle afleveringsopgaver er beskrevet i "Noter, links, software m.m." → "Generel information om opgaver". Formålet er at arbejde med:

- bindinger af værdier og funktioner
- mutérbare variable
- kodedokumentation
- betingelser og løkker

Opgaverne for denne uge er delt i øve- og afleveringsopgaver.

Øve-opgaverne er:

3ø.0 Indtast følgende program i en tekstfil, og oversæt og kørs programmet

Listing 1: Værdibindinger.

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

Forklar hvad parenteser 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 parenteser?

3ø.1 Listing 1 benytter F#'s letvægtssyntaks (Lightweight syntax). Omskriv programmet (enten med eller uden `y` bindingen), så det benytter regulær syntaks.

3ø.2 Følgende program,

Listing 2: Streng.

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

skulle skrive “Hello Jon Sparring!” 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.

3ø.3 Tilføj en funktion

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

til Listing 1, hvor a , b og x er argumenter til udtrykket $ax + b$, og modifier kaldet til `printfn` så det benytter funktionen istedet for udtrykket $(a * x + b)$.

3ø.4 Brug funktionen udviklet i Opgave 3ø.3, således at du udskriver værdien af funktionen for $a = 3$, $b = 4$ og $x = 0 \dots 5$ ved brug af 6 `printfn` kommandoer. Modifier nu dette program vha. af en `for` løkke og kun en `printfn` kommando. Gentag omskrivningen men nu med en `while` løkke.

3ø.5 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.

3ø.6 Fakultetsfunktionen kan skrives som,

$$n! = \prod_{i=1}^n i = 1 \cdot 2 \cdot \dots \cdot n \quad (1)$$

(a) Skriv en funktion

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

som benytter en `while` løkke, en tællevariablen og en lokal variable til at beregne fakultetsfunktionen.

(b) Hvad er det største n , som funktionen kan beregne fakultetsfunktionen for, og hvad er begrænsningen? Lav en ny version,

```
fac : n:int -> int64
```

som benytter `int64` istedet for `int` til at beregne fakultetsfunktionen. Hvad er nu det største n , som funktionen kan beregne fakultetsfunktionen for?

Afleveringsopgaven er:

3i.0 Betragt følgende sum af heltal,

$$\sum_{i=1}^n i. \quad (2)$$

Man kan ved induktion vise, at $\sum_{i=1}^n i = \frac{n(n+1)}{2}$, $n \geq 0$. Opgaven har følgende delafleveringer:

(a) Skriv en funktion

```
sum : n:int -> int
```

som ud over tælleverdien benytter en lokal variabel `s` og en `while` løkke til at udregne summen $1 + 2 + \dots + n$.

(b) Lav en funktion

```
simpleSum : n:int -> int
```

som i stedet benytter formelen $\frac{n(n+1)}{2}$.

(c) Lav et program, som skriver en tabel ud på skærmen med 3 kolonner: `n`, `sum n` og `simpleSum n`, og et passende antal rækker. Verificer ved hjælp af tabellen at de 2 funktioner beregner til samme resultat.

(d) Hvad er det største n de 2 versioner kan beregne sum funktionen korrekt for? Hvordan kan programmet modificeres, så funktionen kan beregnes for større værdier af n ?

3i.1 Som en variant af Opgave 3ø.5, skal der arbejdes med funktionen

```
mulTable : n:int -> string
```

som tager 1 argument og returnerer en streng indeholdende de første $1 \leq n \leq 10$ linjer i multiplikationstabellen inklusiv ny-linje tegn, således at hele tabellen kan udskrives med et enkelt `printf "%s"` statement. F.eks. skal kald til `mulTable 3` returnere

Listing 3: Eksempel på brug og output fra `mulTab`.

```
1 printf "%s" (mulTab 3);;
2      1   2   3   4   5   6   7   8   9  10
3      1   1   2   3   4   5   6   7   8   9  10
4      2   2   4   6   8  10  12  14  16  18  20
5      3   3   6   9  12  15  18  21  24  27  30
```

hvor alle indgange i tabellen har samme bredde. Opgaven har følgende delafleveringer:

(a) Lav

```
mulTable : n:int -> string
```

så den som lokal værdibinding benytter en og kun en streng, der indholder tabellen for $n = 10$, og benyt streng-indicering til at udtrække dele af tabellen for $n < 10$. Afprøv `mulTable n` for $n = 1, 2, 3, 10$.

(b) Lav

```
loopMulTable : n:int -> string
```

så den benytter en lokal streng-variabel, som bliver opbygget dynamisk vha. 2 `for` løkker og `sprintf`. Afprøv `loopMulTable n` for $n = 1, 2, 3, 10$.

- (c) Lav et program, som benytter sammenligningsoperatoren for strenge "=", og som skriver en tabel ud på skærmen med 2 kolonner: `n`, og resultatet af sammenligningen af `mulTable n` med `loopMulTable n` som `true` eller `false`.
- (d) Forklar forskellen mellem at benytte `printf "%s"` og `printf "%A"` til at printe resultatet af `mulTab`.

Afleveringen skal bestå af en zip-fil, som skal indeholde en mappe med fsharp koden. Der skal være en fsharp tekstfil per fsharp-opgave, og de skal navngives `3i0.fsx` osv. De skal kunne oversættes med `fsharpc` og køres med `mono`. Funktioner skal dokumenteres ifølge dokumentationsstandarden, og udover selve programteksten skal besvarelserne indtastes som kommentarer i de `fsx`-filer, de hører til.

God fornøjelse.