Introduktion til programmering, ugeseddel 1

Version 1.2

3. september, 2013

Velkommen til kurset "Introduktion til programmering".

Dette er den første af i alt 7 *ugesedler*. Ugesedlerne indeholder information om ugens opgaver samt diverse praktisk information og perspektivering af ugens pensum.

Den første undervisningsuge har til formål at gøre dig fortrolig med redigering og afvikling af programmer med Emacs og Moscow ML (mosml).

Der er ingen obligatoriske opgaver i uge 1, men der er stillet en gruppeopgave og en individuel opgave som træning til de efterfølgende obligatoriske opgaver. Det anbefales stærkt, at aflevere begge opgaver. Der gives feedback til jeres afleveringer, som kan være nyttig til de senere opgaver.

Vi bruges følgende lærebøger:

- Hansen & Rischel: *Introduction to Programming using SML*, Addison-Wesley 1999. Vi forkorter denne som "HR".
- Nils Andersen: *Supplerende noter i Introduktion til Programmering*, DIKU 2007–2010. Forkortes som "IP-2".

Vi antager at du allerede har anskaffet dig bøgerne ved kursusstart. De kan begge købes i Polyteknisk Boghandel i Biocenteret.

1 Pensum og plan for ugen

Pensum for uge 1 er: HR kap. 1 og 2, IP-2 kap. 1 og 2 (1.7 dog kun kursorisk).

Foreslået læserækkefølge er IP-2, kap. 1, HR kap. 1 og 2, og dernæst IP-2 kap. 2.

Forelæsningen mandag vil fokusere på at skrive simple udtryk og funktioner i emacs og køre dem i mosml. Hyppigt forekomne fejlmedelelser vil blive forklaret. Øvelserne vil give hjælp til installering af Eduroam, Emacs og mosml til de studerende, der ikke har gjort det allerede, og der vil derefter laves simple funktioner i mosml.

Tirsdag omhandler både forelæsninger og øvelser funktionserklæringer, betingede udtryk og simple typer, herunder heltalsaritmetik.

Fredag introducerer rekursive funktioner og billeder. Øvelserne bruges til at arbejde med den individuelle opgave.

2 Mandagsopgaver

1M1 Følg din instruktors anvisninger til installering af mosml, og Emacs på din bærbare. Hvis du ikke har en med selv, så kig med hos en anden.

1M2 Opsæt (med hjælp fra instruktor) Eduroam på din bærbare.

1M3 Log ind på KUnet og find Absalonsiden for IP.

1M4 Start mosml.

1M5 Evaluer udtrykkene `2+3`, `2.0+3.0` og `2+3.0` i mosml og bemærk forskellene mellem resultaterne. Hvad skyldes de?

1M6 Evaluer udtrykket `2-3`. Hvordan ser fortegnet på resultatet ud?

1M7 Evaluer udtrykkene `~3␣-␣~5` og `~3␣-~5` (bemærk forskellen på mellemrumstegn). Hvad sker der?

1M8 Evaluer udtrykket `9*9`, og gentag derefter evaluering af udtrykket `it*it` tre gange. Hvad sker der, og hvorfor?

1M9 Evaluer udtrykket `99.0 * 99.0`, og gentag derefter evaluering af udtrykket `it*it` syv gange. Hvad sker der, og hvorfor?

1M10 Evaluer i mosml udtrykkene `2<3`, `2=3`, `3<=3`, `3>=3` og `3>3`.

1M11 Skriv i Emacs en fil `plus5.sml`, der indeholder en funktionserklæring til en funktion `plus5`, sådan at funktionskaldet `plus5(`n`)` evalueres til $n + 5$ for alle heltal n. F.eks. skal kaldet `plus5(7)` evaluere til værdien `12`. Hent funktionen ind i mosml med kommandoen `use "plus5.sml";`. Hvilken type har funktionen? Hvad sker der, når du laver kaldene `plus5(7)`, `plus5 7`, `plus5(7.0)` og `plus5("7")`?

1M12 Hvis der er tid tilovers, så regn opgave 2.1–2.9 i IP-2.

3 Tirsdagsopgaver

Det forventes, at du inden øvelserne tirsdag har forberedt dig på opgaverne ved at løse så mange som muligt på egen hånd.

1T1 Erklær fakultetsfunktionen (n!) i en fil:

```
fun fact 0 = 1
  | fact n = n * fact(n-1)
```

Hent funktionen ind i mosml og evaluer kaldene `fact 0`, `fact 7` og `fact 25`. Forklar resultatet af det sidste kald.

Evaluer kaldet `fact(~3)`. Hvad sker der, og hvorfor?

1T2 Erklær potensfunktionen power i en fil:

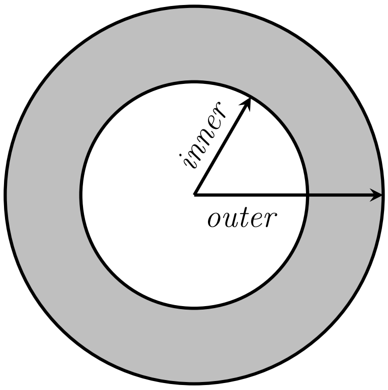
```
fun power (x, 0) = 1
  | power (x, n) = x * power (x,n-1)
```

Evaluer kaldene `power (2,4)` og `power (2.0,4)`. Hvad sker der og hvorfor?

1T3 Overvej, hvorfor funktioner og værdier har typer.

1T4 Håndkør beregningen af `power (3,3)`, dvs. skriv de forskellige skridt i beregningen ned.

1T5 Erklær i en fil en funktion `ringArea : real * real -> real`, sådan at kaldet `ringArea(`$outer$`, `$inner$`)` er arealet af en ring med radius $outer$ og et hul med med radius $inner$. Det vil sige, det grå areal i følgende figur:



Vink: brug funktionen `circleArea` fra HR afsnit 1.2.

1T6 Erklær i en fil en funktion `fooint : int * int -> int`, sådan at kaldet `fooint(`n`,`k`)` evaluerer til værdien $2n - k^2$ for heltal n og k. Beregn `fooint(3,2)`.

Erklær dernæst en funktion `fooreal : real * real -> real`, sådan at kaldet `fooreal(`n`,`k`)` evaluerer til værdien $2n - k^2$ for kommatal n og k. Vær opmærksom på to-tallet. Beregn `fooreal(3.0,2.0)`.

Erklær dernæst en funktion `foomix : real * int -> real`, sådan at kaldet `foomix(`n`,`k`)` evaluerer til værdien $2n - k^2$ for kommatal n og heltal k. Bemærk de forskellige typer af argumenterne til minus. Hvordan håndterer du det? Beregn `foomix(3.0,2)`.

1T7 Hvis der er tid tilovers, løs opgave 1.3, 1.4, 2.1, 2.2 i HR samt 4.1–4.6 i IP-2.

4 Gruppeaflevering

I uge 1 er gruppeafleveringen frivillig, men det anbefales at aflevere den og få feedback. Besvarelsen afleveres i Absalon. Der afleveres en fil pr. gruppe, men den skal angive alle deltageres fulde navne i kommentarlinjer øverst i filen. Filens navn skal være af formen `1G-`*`initialer`*`.sml`, hvor initialer er erstattet af gruppemedlemmernes initialer. Hvis f.eks. Bill Gates, Linus Torvalds og Steve Jobs afleverer en opgave sammen, skal filen hedde `1G-BG-LT-SJ.sml`. Brug gruppeafleveringsfunktionen i Absalon.

I gymnasiet lærte I, at man kan løse andengradsligningen $ax^2+bx+c = 0$ med formlen

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Som giver 2 reelle løsninger, hvis $b^2 - 4ac \geq 0$ og ingen reelle løsninger, hvis $b^2 - 4ac < 0$.

1G1 Skriv en funtion `solve2 : real * real * real -> real * real`, sådan at `solve2(`a`,`b`,`c`)` giver de to løsninger for x i ligningen $ax^2 + bx + c = 0$, såfremt $b^2 - 4ac \geq 0$. Du behøver ikke tage stilling til tilfældet $b^2 - 4ac < 0$. Vink: Kvadratrodsfunktionen `sqrt` findes i biblioteksmodulet `Math`.

1G2 Hvad giver kaldet `solve2(2.0,3.0,1.0)`? Er svaret rigtigt?

1G3 Hvad giver kaldet `solve2(2.0,3.0,4.0)`? Hvorfor?

Den potensfunktion `power`, der er defineret i tirsdagsopgaverne skal bruge 21 multiplikationer til at beregne `power(2,21)`–helt generelt bruges n multiplikationer til at beregne `power(`a`,`n`)`. Det kan gøres med væsentligt færre multiplikationer ved at benytte regnereglen $a^{2n} = (a^n)^2$, når man har en lige potens.

1G4 Skriv en ny potensfunktion `powerNew : int * int -> int`, som bruger denne regneregel til at reducere antallet af multiplikationer, så f.eks. `powerNew(2,21)` kan beregnes med højest 11 multiplikationer. Vink: Brug funktionerne `div` og `mod`.

1G5 Udvid funktionen, så den udover an også returnerer antallet af brugte multiplikationer. Lav altså en funktion `powerCount : int * int -> int * int`, sådan at `powerCount(`a`,`n`)` returnerer parret `(`a^n`,`m`)`, hvor m er antallet af brugte multiplikationer. Vis resultatet for `powerCount(2,21)`.

5 Individuel aflevering

Også den individuelle opgave er frivillig i uge 1, men vi anbefaler igen, at den afleveres. Den afleveres i Absalon som en fil med navnet `1I-`*`navn`*`.sml`, hvor *navn* er erstatttet med dit navn. Hvis du fx hedder Anders A. And, skal filnavnet være `1I-Anders-A-And.sml`. Skriv også dit fulde navn som en kommentar i starten af filen.

1I1 Lav en funktion `powerRealInt: real * int -> real`, der givet et kommatal a og et heltal n beregner a^n. Funktionen skal virke både for positive og negative værdier af n. Brug reglen $a^{-n} = \frac{1}{a^n}$. Vis værdien af kaldet `powerRealInt (0.99,~100)`.

1I2 To heltal p og q (hvor, for enkelthed, vi kan antage at $1 < p < q$) siges at være indbyrdes primiske (*relative primes*), hvis det eneste tal, der går op i både p og q er 1. Lav en funktion `relativePrimes : int * int -> bool`, sådan at `relativePrimes(`p`,`q`)` er `true` hvis og kun hvis p og q er indbyrdes primiske. Vink: brug funktionen `gcd` fra afsnit 2.3 i HR som inspiration.

1I3 Lav en funktion `nextNotRelativePrime : int -> int`, som givet et heltal $n > 0$ finder det mindste heltal $m > n$ sådan at m og n ikke er indbyrdes primiske. Fx skal `nextNotRelativePrime 6` give 8, da 6 og 7 er indbyrdes primiske, mens 6 og 8 begge har 2 som divisor. `nextNotRelativePrime 7` skal returnere 14, da alle tallene 8, 9, 10, 11, 12 og 13 er indbyrdes primiske med 7, mens 7 og 14 har 7 som fælles divisior. Vis resultatet af `nextNotRelativePrime 119`.

6 Ugens nød

1N1 Lav en funktion `tree : int -> string`

således at `tree` n laver et træ af stjerne-tegn i n lag. Fx giver `tree 4` strengen `"␣␣␣*␣␣␣\n␣␣***␣␣\n␣*****␣\n*******\n"`.

Det vil sige, hvis du udfører

```
print(tree 5);
```

i toplevel, så skal du se noget i stil med:

```
    *
   ***
  *****
 *******
*********
```

1N2 Lav en funktion

```
treeline : int * int -> string
```

så `treeline(`m`, `n`)` laver en linje med m træer, hvert med n lag. Det vil sige, at hvis du udfører

```
print(treeline(4, 5));
```

i toplevel, så skal du se noget i stil med:

```
    *         *         *         *
   ***       ***       ***       ***
  *****     *****     *****     *****
 *******   *******   *******   *******
********* ********* ********* *********
```

1N3 Lav en funktion

```
forest : int * int * int -> string
```

så `forest(`r`, `m`, `n`)` laver r rækker af m træer, hvert med n lag.

Det vil sige, hvis du udfører

```
print(forest(3, 4, 5));
```

i toplevel, så skal du se noget i stil med:

```
    *         *         *         *
   ***       ***       ***       ***
  *****     *****     *****     *****
 *******   *******   *******   *******
********* ********* ********* *********
    *         *         *         *
   ***       ***       ***       ***
  *****     *****     *****     *****
 *******   *******   *******   *******
********* ********* ********* *********
    *         *         *         *
   ***       ***       ***       ***
  *****     *****     *****     *****
 *******   *******   *******   *******
********* ********* ********* *********
```