Ordinær eksamen i Funktionsprogrammering, blok 1 2007

9. oktober 2008

Nærværende dokument udgør eksamensopgaven for den ordinære eksamen i kurset "Funktionsprogrammering", blok 1 2007.

Dokumentet offentliggøres onsdag 24. oktober kl. 17 via KUs kursusadministrationssystem Absalon. Eksamensbesvarelsen skal afleveres senest fredag 26. oktober kl. 17 — se afsnit 1 nedenfor.

Eksamensbesvarelsen evalueres som bestået/ikke-bestået. Eksamensresultaterne vil findes på Absalon senest tre uger efter eksamens afslutning, dvs. 16. november 2007 og vil findes i det Naturvidenskabelige Fakultets eksamensprotokol umiddelbart derefter.

Eksamenen består af 10 opgaver, der vægtes ligeligt. Eksamen vurderes bestået, hvis besvarelsen vurderes til at have løst 50% eller mere af sættet (bemærk, at delvist færdige besvarelser tæller med — man bør således aflevere sin besvarelse *under alle omstændigheder*).

Eksamensopgaven skal besvares *individuelt*. Ethvert samarbejde om opgaven er forbudt og vil blive behandlet i henhold til fakultetets gældende regler om eksamenssnyd. Se dog afsnit 2 nedenfor.

Hvis der opstår tvivl om selvstændighed eller kvalitet i besvarelsen, kan studerende blive indkaldt til en supplerende mundtlig eksamen torsdag 8. november. Studerende vil blive indkaldt til mundtlig eksamen per brev til deres KU-konti onsdag 7. november kl. 12.00, hvorfor alle studerende bedes efterse deres konti på denne dato. Udeblivelse fra mundtlig eksamen vil resultere i karakteren ikke-bestået eller, hvis der er tale om indkaldelse grundet mistanke om eksamenssnyd, i indberettelse af den studerende til dekanen.

I tilfælde af uklarheder i opgaveteksten er det op til eksaminanderne selv at specificere opgaven; se dog afsnit 2 nedenfor.

1 Vejledning i aflevering

Besvarelsen skal afleveres elektronisk via Absalon senest fredag 26. oktober kl. 17.00 efter følgende procedure:

På kursets Absalon-hjemmeside findes menupunktet "Eksamen", hvorunder et opgavepunkt forefindes med titlen "Aflevering af eksamen". Under dette punkt skal besvarelsen af eksamensopgaver uploades efter samme procedure som aflevering af de obligatoriske opgaver på kurset.

Alle opgaverne skal afleveres i *en* fil navngivet "efternavn.fornavn.sml". Hedder man f.eks. "Jakob Simonsen", skal filen således navngives "Simonsen.Jakob.sml". Således skal besvarelse af opgaver, der kræver tekst, der ikke kan afvikles i MosML, angives som ML-kommentarer, dvs. `(* ... *)`.

Filens indhold skal kunne afvikles under MosML 2.01 på DIKUs system vha. kommandoen `mosml -P full`.

Det er afgørende, filens indhold kan afvikles samt at funktioner i filen har de navne og typer, der er specificeret i opgaveteksten; i modsat fald kan man risikere, at hele eksamensbesvarelsen vil blive betragtet som ukorrekt. Hvis man har en delvis løsning til en delopgave, som ikke kan afvikles, må denne indsættes i ML-kommentarer `(* ... *)`.

I opgaver, hvor man bliver bedt om at skrive tekst (f.eks. forklaringer), der ikke kan afvikles under MosML, må denne tekst tillige indsættes i ML-kommentarer.

Alle funktioner i besvarelsen forventes kommenteret iht. til god kommentarskik, se evt. FP-2 afsnit 5.3.3, og al programtekst skal opstilles pænt med passende indrykning.

Modsat de obligatoriske opgaver kræves der *ikke* afprøvning af ML-funktionerne, som udarbejdes i forbindelse med eksamensbesvarelsen. De studerende forventes og skyndes til at afprøve deres funktioner selv for at sikre korrekte besvarelser.

Bemærk, at det er muligt for studerende at uploade mere end en fil. Den *senest rettidigt afleverede* fil — og kun den — vil blive anset for eksamensbesvarelsen.

Skulle der mod forventning opstå tekniske problemer med Absalon i tidsrummet umiddelbart før afleveringsfristen for besvarelsen, kan studerende konsultere `http://www.diku.dk/~simonsen`, hvor et opslag med instruktioner vil blive lagt op i dette tilfælde.

2 Støtte i eksamensperioden

De studerende skyndes til at benytte kursets diskussionsforum på Absalon til spørgsmål eller diskussion af uklarheder i opgaveteksten samt til spørgsmål om formalia i forbindelse med eksamen. Decideret diskussion af opgavens faglige indhold, herunder forslag til løsninger af eksamensopgaverne må ikke forekomme på forummet.

Spørgsmål kan tillige rettes personligt til den kursusansvarlige, Jakob Grue Simonsen, N107, tlf. 35 32 14 39, e-mail: simonsen@diku.dk gennem hele eksamensperioden. De studerende opfordres dog til at stille spørgsmål, hvis svar kunne have interesse for andre studerende, på forummet.

3 Eksamensopgaver

Eksamensopgaverne nedenfor vægtes ligeligt. En grov vurdering af opgavernes sværhedsgrad i stigende rækkefølge er: Opgave 1, 2, 7, 6, 3, 4, 8, 9, 5, 10

Det understreges, at sværhedsgraden ovenfor er en subjektiv vurdering foretaget af kursets lærere. Studerende kan have en anden opfattelse.

Nedenstående information benyttes i opgaverne 1–5

For et helt positivt tal m (de positive heltal er $\{1, 2, 3, \ldots\}$) vil vi ved en m-*liste* forstå en liste med m elementer og ved et m-*kvadrat* en m-liste af m-lister. Erklæres for eksempel

```
val kvdr = [[1, 4, 7], [6, 0, 2], [3, 8, 5]]
```

så er `kvdr` et 3-kvadrat.

Til behandling af m-kvadrater i SML erklæres

```
type 'a kvadrat = 'a list list;
```

Nu gælder `kvdr : int kvadrat`. (Vær opmærksom på, at MosML opfatter `'a kvadrat` som en *forkortelse* og i sine systemsvar benytter den ordinære beskrivelse `'a list list`.)

Opgave 1 Erklær en undtagelse `KFejl` og en funktion `rangK : 'a kvadrat -> int`, sådan at hvis *xss* er et m-kvadrat for et eller andet tal m, så vil `rangK` *xss* have værdien m, mens `rangK` *xss* vil hejse undtagelsen `KFejl`, hvis *xss* ikke er et kvadrat.

Der skal for eksempel gælde `rangK kvdr = 3`, mens `rangK []`, `rangK [[]]` og `rangK [["ab", "c"]]` alle skal kaste `KFejl`.

Opgave 2 Erklær en funktion `mapK : ('a -> 'b) -> 'a kvadrat -> 'b kvadrat`, sådan at `mapK` *f xss* vil danne det nye kvadrat, som opstår ved at anvende *f* på hvert enkelt af felterne i *xss*.

Defineres for eksempel

```
fun stjerner 0 = ""
  | stjerner n = "*" ^ stjerner (n - 1)
```

så skal `mapK stjerner kvdr` blive
`[["*", "****", "*******"], ["******", "", "**"], ["***", "********", "*****"]]`.

Opgave 3 Et m-kvadrat kan opfattes som en repræsentation af en to-dimensional opstilling med m rækker og m søjler. Mere præcist kan man opfatte værdien

$$xss = [[x_{0,0}, x_{0,1}, \ldots, x_{0,m-1}], [x_{1,0}, x_{1,1}, \ldots, x_{1,m-1}], \ldots, [x_{m-1,0}, x_{m-1,1}, \ldots, x_{m-1,m-1}]]$$

som en repræsentation af

$$\begin{array}{cccc} x_{0,0} & x_{0,1} & \cdots & x_{0,m-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m-1,0} & x_{m-1,1} & \cdots & x_{m-1,m-1} \end{array}$$

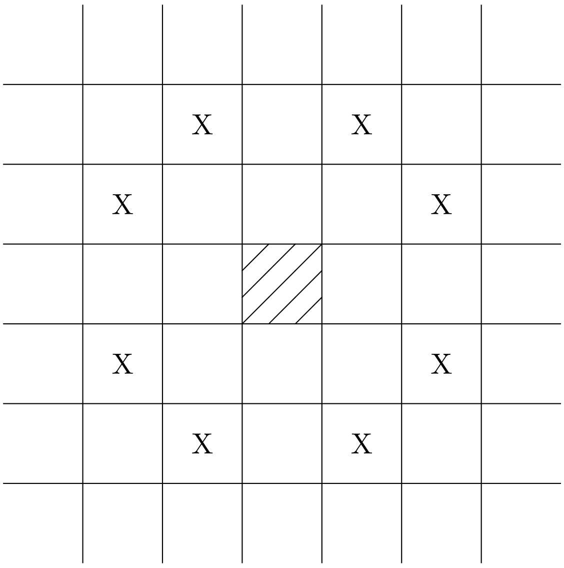
Erklær en funktion `visK : string kvadrat -> string`, sådan at `print (visK` *xss*`)` vil udskrive en todimensional opstilling af m-kvadratet *xss*. Hvis ikke alle de m^2 indgående tekster i *xss* er lige lange, skal de før udskrivningen forlænges (i højre side) med blanktegn, så de alle får samme længde som den længste indgående tekst. Består den længste indgående tekst af t tegn, kommer hver linje i udskriften altså ud over linjeskiftet til at indeholde $m \cdot t$ tegn.

For `stjernekvdr = mapK stjerner kvdr` fra opgave 3 skal `print (visK stjernekvdr)` for eksempel danne

```
*       ****    *******
******          **
***     *************
> val it = () : unit
```

Opgave 4 I et m-kvadrat nummereres rækker og søjler fra 0 til $m-1$, så et felt kan angives ved sin *position*, det vil sige ved et par (i, j) bestående af række- og søjlenummer.

Afstanden mellem to felter siges at være *et springertræk*, hvis rækkeafstanden er 1 og søjleafstanden er 2 eller omvendt rækkeafstanden er 2 og søjleafstanden 1. Hvis et felt ikke ligger for tæt på kanter eller hjørner, vil det have 8 andre felter i et springertræks afstand (se figuren).



Konstruer en funktion
`springernaboer : int -> int * int -> (int * int) list`, sådan at `springernaboer` m (i, j) for feltet i position (i, j) i et m-kvadrat danner listen af positioner på felter, der befinder sig et springertræk fra det (og ligger inden for kvadratets kanter).

Opgave 5 Ved en *springertur* på et m-kvadrat forstås et m-kvadrat af type `int kvadrat`, hvori nogle af felterne er nummereret $1, 2, \ldots, n$ ($n \geq 1$) på en sådan måde, at 1 står i et af hjørnerne, og så der er et springertræk mellem felt 1 og felt 2, et springertræk mellem felt 2 og felt 3, og så videre, et springertræk mellem felt $n - 1$ og felt n. Tallet $n \geq 1$ kaldes for springerturens *længde*. I de $m^2 - n$ resterende felter står der 0.

Figuren viser `kvdr`, som er en springertur (af længde 8) på et 3-kvadrat:

1	4	7
6	0	2
3	8	5

Skriv en funktion `tjekSpringertur : int kvadrat -> bool`, der netop bliver `true`, hvis argumentet er en springertur.

Opgave 6 Betragt følgende funktionserklæring

```
local
  fun getLast [x] = x
    | getLast (x::xs) = getLast xs
in
  fun ineffMul [] = 0
    | ineffMul (xr as x::xs) = getLast xr + ineffMul xs
end;
```

Forklar kort, hvad getLast beregner, og angiv køretidskompleksiteten for getLast i Θ- eller $\mathcal{O}$-notation. Argumenter kort for svaret.

Forklar dernæst kort, hvad ineffMul beregner, og angiv køretidskompleksiteten for ineffMul i Θ- eller $\mathcal{O}$-notation. Argumenter kort for svaret. Vink: Nærlæs kapitel 7 i FP-2.

Opgave 7 Betragt følgende erklæring

```
fun lessEqual n = fn i => i <= n;
```

Giv en kort og klar beskrivelse af, hvad `lessEqual` beregner (hvad er værdien af `lessEqual n`?).

Erklær en funktion `mapLessEqual : int list -> (int -> bool) list`, som givet en liste $[n_1, n_2, ...n_i, ..., n_k]$ af heltal, returnerer en liste af funktioner $[f_1, f_2, ..., f_i, ...f_k]$, således at alle funktionerne $f_1, f_2, \ldots, f_k$ har typen `int -> bool`, og der for funktionerne gælder:

$f_1(j) = \textbf{true}$ hvis og kun hvis $j \leq n_1$
$f_2(j) = \textbf{true}$ hvis og kun hvis $j \leq n_2$
$\vdots$
$f_k(j) = \textbf{true}$ hvis og kun hvis $j \leq n_k$.

Eksempel på kald:

```
- val list1 = mapLessEqual [4,9];
> val list1 = [fn, fn] : (int -> bool) list
- val func1 = hd list1;
> val func1 = fn : int -> bool
- func1 5;
> val it = false : bool
- func1 3;
> val it = true : bool
```

Vink: Funktionserklæringen kan skrives meget kort.

Opgave 8 Skriv en funktion

`opdel : string -> string list list`,

der finder alle mulige opdelinger af en tekst i ikke-tomme deltekster. Rækkefølgen af resultaterne er underordnet. F.eks. kunne kaldet `opdel "abc"` returnere listen `[["a", "b", "c"], ["a", "bc"], ["ab", "c"], ["abc"]]`.

Opgave 9 Det er et klassisk resultat, at ethvert positivt heltal på entydig måde kan skrives som et (evt. tomt) produkt af primtal. (NB: 1 er *ikke* et primtal.) F.eks. er $120 = 2^3 \times 3 \times 5$.

Skriv en funktion

`faktoriser : int -> (int * int) list`,

så det for ethvert $n \geq 1$ gælder, at kaldet `faktoriser` n returnerer en liste $[(p_1, m_1), \ldots, (p_k, m_k)]$ af heltalspar, hvor $k \geq 0$, hvert p_i er et primtal, $p_1 < \cdots < p_k$, hvert $m_i \geq 1$, og $p_1^{m_1} \times \cdots \times p_k^{m_k} = n$. F.eks. skal kaldet `faktoriser 120` returnere listen `[(2, 3), (3, 1), (5, 1)]`. Funktionens opførsel for $n \leq 0$ er underordnet.

Der stilles ingen særlige krav til `faktoriser`s køretid, men den bør være "rimelig": Ethvert heltal repræsenterbart i Moscow ML (dvs. mindre end 2^{30}) bør kunne faktoriseres på højst et par sekunder.

Opgave 10 Givet en ordningsrelation $\leq_t$ mellem elementer af en type t kan vi definere dens *leksikografiske udvidelse*, $\leq_{t\ \mathtt{list}}$, mellem elementer af typen t `list`, analogt til leksikografisk ordning af tekster ud fra ordningen på de enkelte tegn: En liste er leksikografisk mindre end eller lig en anden, netop når deres elementer startende fra venstre er parvis ens, indtil den første liste enten er færdig eller indeholder et element strengt mindre end det tilsvarende element i den anden liste.

Hvis $\leq_{\mathtt{int}}$ f.eks. er den sædvanlige aritmetiske ordning på heltal, gælder `[3,1]` $\leq_{\mathtt{int\ list}}$ `[3,1]`, `[2,5]` $\leq_{\mathtt{int\ list}}$ `[2,5,3]` og `[7,2]` $\leq_{\mathtt{int\ list}}$ `[7,3]`. Symbolsk kan ordningen udtrykkes som følger:

$$[a_1, \ldots, a_n] \leq_{t\ \mathtt{list}} [b_1, \ldots, b_m] \iff \exists k \geq 0.\, a_1 =_t b_1 \wedge \ldots \wedge a_k =_t b_k \wedge (k = n \vee a_{k+1} <_t b_{k+1}),$$

hvor $x =_t y$ betyder $x \leq_t y \wedge y \leq_t x$, og $x <_t y$ betyder $\neg(y \leq_t x)$.

En type med en ordningsrelation kan i SML skrives som en struktur med følgende signatur:

```
signature ORD =
sig
  type t
  val leq : t * t -> bool
end
```

F.eks. kan vi definere en `ORD`-struktur ud fra den indbyggede (overlæssede) sammenligningsoperator `<=` for tegn:

```
structure OrdChar : ORD =
struct
  type t = char
  val leq = op <=
end
```

Skriv en funktor `Lex` med hovedet

```
functor Lex (structure O : ORD) :
  ORD where type t = O.t list =
struct
  (* skriv mig! *)
end
```

så ordningsrelationen i resultatstrukturen netop er den leksikografiske udvidelse af argumentstrukturens. Specielt bør det gælde, at efter erklæringerne

```
structure LexChar = Lex (structure O = OrdChar)
fun leqs (s1,s2) = LexChar.leq (explode s1, explode s2);
```

vil `leqs : string * string -> bool` give samme resultater som MLs overlæssede sammenligningsoperator `<=` for typen `string`.

(Eksamenssættet slut)