UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i: INF3110/4110 - Programmeringsspråk

Eksamensdag: 1. desember 2005

Tid for eksamen: 14:30 – 17:30

Oppgavesettet er på 8 sider inklusiv side 8 som brukes til besvarelse

Vedlegg: Ingen

Tillatte hjelpemidler:Alle

Kontroller at oppgavesettet er komplett før du begynner å besvare spørsmålene.

Dette oppgavesettet består av 3 oppgaver som kan løses uavhengig av hverandre. Dersom du synes noe i oppgaveteksten er uklart, må du gjøre dine egne forutsetninger; sørg bare for at disse er tydelig angitt.

Lykke til!

Innhold

Oppgave 1 Runtime-systemer, skoping, typer (vekt 40%)

Del I

NB.: Denne delen av oppgave 1 besvares ved å fylde ut og tegne resten av stakken på side 8 og legge denne ved besvarelsen! Husk å fylle ut kandidatnummer og dato.

Anta at vi har følgende program i et programmeringsspråk med funksjoner som parametre. Uttførelsen starter ved å utføre `main()`.

```
{ int i=1, j=2, k=0;
  void p(){
    k=i+j;
    print(k)
  };
  void q(void f()){
    int i=4, j=5, k=6;
    void r(){
      k = i+j;
      print(k);
    };
    f();
    print (k);
    q(r);
  };
  main(){
    q(p);
  };
}
```

1a

Angi for både statisk og dynamisk scoping hva som skrives ut av de fire første kallene på `print` under kjøring av programmet. Legg merke til at disse fire første kallene ikke alle er av den samme `print`!

1b

Tegn stakken som den ser ut når aktiveringsblokken (activation record) for kallet på `r` er på toppen av stakken (Dette skjer via kallet `q(r)` i `q`). Angi både statisk link (access link) og dynamisk link (control link) for hver aktiveringsblokk.

1c

For å implementere funksjoner som parametre i et statisk skopet språk må det for hver aktuell funksjon overføres en closure. En closure består av en link til en aktiveringsblokk og en link til koden.

I eksemplet over er det 2 eksempler på overføring av funksjoner:

```
(1) q(p) i main
(2) q(r) i q
```

Angi for hver av disse tilfeller closure-linken til den aktuelle aktiveringsblokken ved å tegne dem inn på arket side 8.

Del II

Anta at vi har et programmeringsspråk, hvor vi både har klasser og funksjoner, og hvor det i tillegg til subklasserelasjon mellom klasser også er definert en subtyperelasjon mellom funksjoner. Mens det for klasser er den enkle regel at en subklasse også er en subtype til typen representert ved superklassen, så er subtyping for funksjoner definert slik:

For to funksjoner `f` og `f'` er `f'` en subtype av `f` (skrives `f' < f`) hvis et hvert kall av `f` like gjerne kan erstattes av et kall av `f'`, dvs at hvis følgende setning

```
i=f(j)
```

er typeriktig, da er

```
i=f'(j)
```

også typeriktig.

Felles for spørsmål a) og b) har vi definert klassene Point og ColorPoint. ColorPoint er en subklasse til Point:

```
class Point{...}
class ColorPoint extends(Point){...}
```

1d

Vi har følgende funksjoner:

```
int dist(Point p){...}
int dist'(ColorPoint cp){...}
```

som f.eks. beregner avstanden fra origo (ikke viktig hva de rent faktisk gjør, det viktige er signaturen til funksjonene).

Hvis dette språket skal være statisk typesikkert, hvilket av de to alternativer må da gjelde?

```
1) dist  < dist'
2) dist' < dist
```

Begrunn svaret.

1e

Vi har følgende funksjoner:

```
Point somePoint(int d){...}
ColorPoint somePoint'(int d){...}
```

som f.eks. genererer tilfeldige punkter med avstanden `d` fra origo (ikke viktig hva de rent faktisk gjør).

Hvis dette språket skal være statisk typesikkert, hvilket av de to alternativer må da gjelde?

```
1) somePoint  < somePoint'
2) somePoint' < somePoint
```

Begrunn svaret.

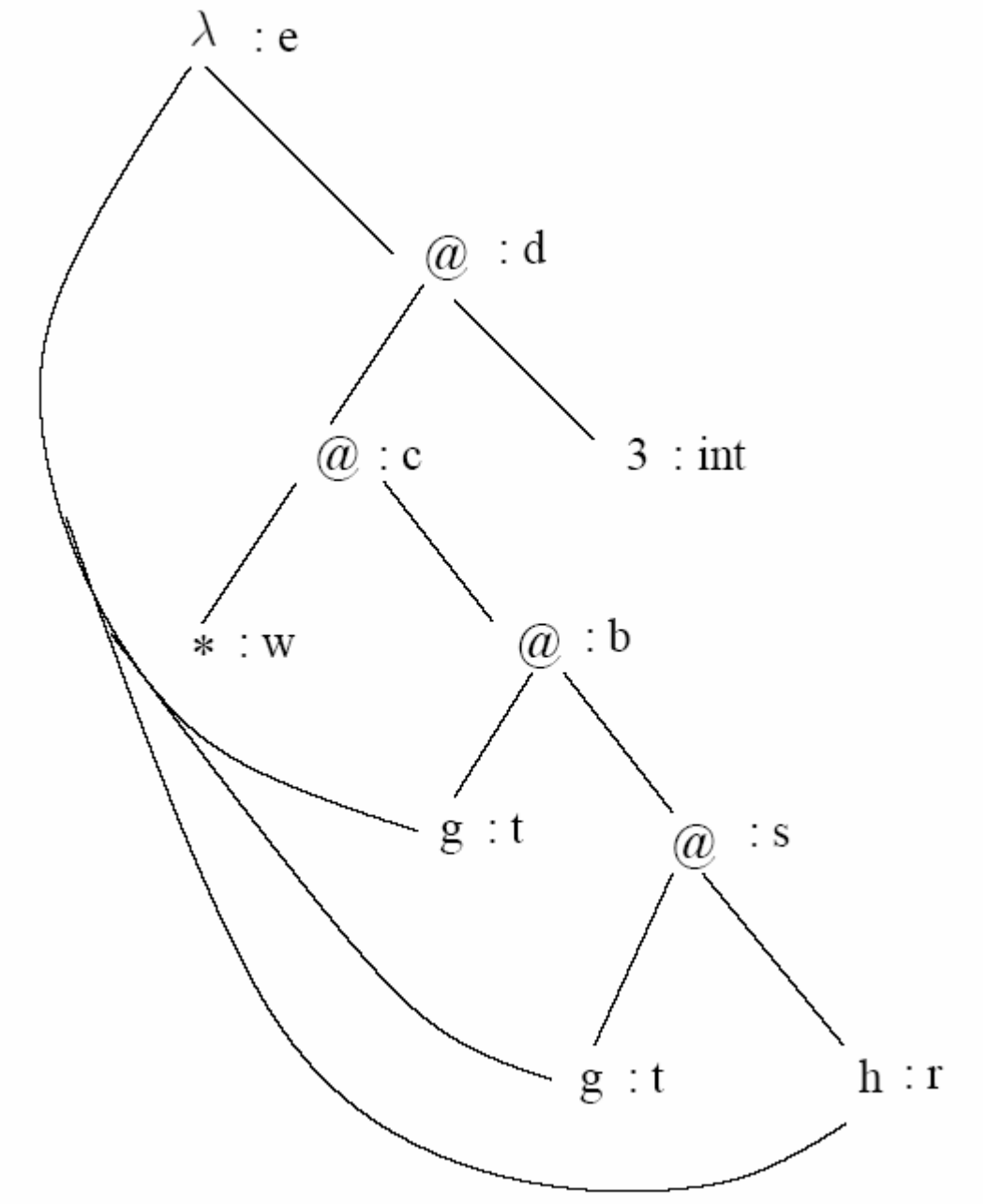
Oppgave 2 ML (vekt 40%)

2a

Avled typen til følgende funksjon:

```
fun f(g,h) = g(g(h)) * 3;
```

Gjør ferdig parsetreet under ved å følge stegene til MLs algoritme for type-inferens (ML type-inference algorithm). Beskriv eksplisitt algoritmens tre steg.



2b

Høyere ordens programmering (higher-order programming) er en av de mer karakteristiske egenskaper ved ML. Noen av MLs høyere ordens funksjoner som opererer på lister er `map`, `foldr` og `foldl`, som er definert slik:

```
fun map f []      = []
  | map f (x::xs) = (f x) :: map f xs;

fun foldr f e []      = e
  | foldr f e (x::xs) = f (x, foldr f e xs);

fun foldl f e []      = e
  | foldl f e (x::xs) = foldl f (f(x, e)) xs;
```

1) Definer, ved å bruke høyere ordens programmering, en funksjon `firstElems`

```
val firstElems = fn : 'a list list -> 'a list
```

som brukt på en liste av lister av integers (int list list) gir en liste som bare inneholder det første elementer fra hver subliste. For eksempel, hvis `a` er definert som følgende liste:

```
val a = [[2,5,76,8], [3,23,45], [5,27,1,43], [34,2,56]];
```

da er resultatet av firstElems(a):

```
- firstElems a;
 > val it = [2,3,5,34] : int list
```

(Hints: Bruk `map` funksjonen. Husk at lister har en oprasjon `hd` som gir første element i en liste, og `tl` som gir resten av listen (tail).)

2) Definer, ved å bruke høyere-ordens programmering, en funksjon `prodFirstElems`

```
val prodFirstElems = fn : int list list -> int
```

som brukt på en liste av lister av integers (int list list) gir produktet av de første elementer i hver subliste. For eksempel, hvis `a` er definert som i 1) ovenfor, da er

```
- prodFirstElems a;
 > val it = 1020 : int
```

fordi 2*3*5*34 = 1020.

(Hints: Bruk `foldr` eller `foldl`. Husk at en infix operator kan transformers til prefix form ved å bruke nøkkelordet `op`; for eksempel, istedet for å skrive `3*5` kan man skrive `op*(3,5)`.)

2c

Vi sier at en funksjon f_1`: int * real -> real` er definert for gitte argumenter `x:int` og `y:real` hvis det er en verdi `z: real` slik at f_1`(x,y)=z`; ellers sier vi at funksjonen er *udefinert* for de gitte argumenter. Vi kan definere følgende tre udefinerte verdier:

- ``uncaught exception'': dette skjer hvis en exception blir 'raised' uten å bli håndtert (handled);
- nan: dette er resultatet som ML gir hvis man for eksempel utfører ``0.0/0.0'';
- inf: dette er resultatet av 'division by zero', for eksempel ``1.0/0.0''.

Vi sier at to funksjoner f_1`: int * real -> real` og f_2`: int * real -> real` er *ekvivalente* hvis og bare hvis (a) når en av funksjonene er definert, da er den andre også definert, og de gir samme resultat (dvs f_1`(x,y)=`f_2`(x,y)`), og (b) funksjonene er udefinert for nøyaktig de samme par

av parameterverdier.

Gitt følgende ML funksjon:

```
exception OddNum;
fun f(0,count) = count
      |  f(1,count) = raise OddNum
      |  f(x,count) = f(x-2, count+1.0) handle OddNum => ~1.0;
```

Skriv en ML funksjon `eqTof` som er ekvivalent med f, uten å bruke exceptions.

Oppgave 3 Prolog (vekt 20%)

3a

Gitt følgende (partielle) database:

```
father(jon,mikael).
father(jon,juan).
father(jon,per).
father(mikael,carlos).
father(per,karl).
father(per,anne).
father(per,sofia).

male(jon).
male(mikael).
male(juan).
male(per).
male(carlos).
male(karl).

female(sarah).
female(anne).
female(sofia).
```

1. Legg til 4 fakta som definerer en `mother` relasjon mellom de tre kvinnelige (female) personer og andre personer (som du kan velge) i databasen ovenfor (f.eks. `mother(sarah,carlos)`).
2. Legg til regler for å definere følgende relasjoner (relationships): `son`, `daughter`, `uncle`, `brother` and `parent`. Noen regler kan gjerne bruke (noen av) `male`, `female`, `mother` and `father`, samt noen av de regler som du definerer her.
3. Skriv to regler `atleastonebrother` og `unclefemale` for å kunne stille spørsmål (queries) for å finne følgende informasjon:
    a. Alle personer som har i det minste (at least) en bror (brother) (bruk `atleastonebrother`)
    b. Alle personer som er onkel (uncle) til en kvinne (female) (bruk `unclefemale`).

3b

Naturlige tall kan defineres slik i Prolog:

```
natural_number(0).
natural_number(s(X)) :- natural_number(X).
```

Det første faktum sier at "`0`" er et naturlig tall, mens det andre sier at hvis `X` er et naturlig tall, så er `s(X)` også et naturlig tall.

Sum og produkt av to naturlige tall kan definers slik:

```
plus(0,X,X) :- natural_number(X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).

prod(0,X,0) :- natural_number(0).
prod(s(X),Y,Z) :- prod(X,Y,XY), plus(XY,Y,Z).
```

Skriv et Prolog program `exp(M,X,Y)` som beregner den (matematiske) eksponering $X^M=Y$. For eksempel, det å skrive spørsmålet (query)

```
exp(s(s(0)), s(s(s(0))), Y).
```

(som representerer $3^2=Y$) vil produsere svaret

```
Y = s(s(s(s(s(s(s(s(s(0)))))))))
```

(som representerer $Y=9$).

Ark til besvarelse av Oppgave 1, Del I

Kandidat nr: .....

Dato: .....

1a

	statisk scoping	dynamisk scoping
1. utførelse av en print(k)		
2. utførelse av en print(k)		
3. utførelse av en print(k)		
4. utførelse av en print(k)		

1 b & c