NTNU

Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i **TD4165 Programmeringsspråk**

Faglig kontakt under eksamen: Øystein Nytrø

Tlf: 91897606

Eksamensdato: 6. august 2015

Eksamenstid (fra–til): 09.00 – 13.00

Hjelpemiddelkode/Tillatte hjelpemidler: C: Ingen trykte og håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Annen informasjon:
Oppgaven har 4/19 deloppgaver/punkter. Alle har samme vekt. Skriv tydelig, kort og presist.

Alle programmer skal skrives i Oz.

Målform/språk: bokmål

Antall sider: 4

Antall sider vedlegg: 0

Kontrollert av:

Dato Sign

Merk! Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

Oppgave 1 Innledende om programmeringsspråk (6 punkter)

a) Hva er et dataflytspråk? Er Oz et dataflytspråk? Hvorfor?

b) Hvilke egenskaper har Java som gjør det ikke-deklarativt?

c) Fjern syntaktisk sukker fra Oz-uttrykkene:

```
[2 4 6 8]
2|4|6|8|nil
2|(4|(6|(8|nil)))
2#4
2#4#6#8
"2468"
```

d) Hva er forskjellen på post-typer (records) og tupler i Oz? Hva er forskjellen på tupler og lister? Lag lite Oz-program som henter det tredje elementet i en liste (med minst tre elementer) uten iterasjon eller rekursjon.

e) Hva er resultatet av å utføre følgende program?

```
declare T I Y LT RT W in
T = tre(nkl:I val:Y ven:LT hoey:RT)
I = adams Y = 42 LT = nil RT = nil W = tre(I Y LT RT)
{Browse [T W]}
```

f) Funksjonene `BinTre` og `BinTre2` gitt under har samme mening. Hvilken? Hvilken viktig forskjell er det i effektiviteten av av funksjonene? Forklar med et eksempel.

```
local
   proc {Og B1 B2 ?B}
      if B1 then B = B2 else B = false end
   end
in
   proc {BinTre T ?B}
      case T
      of nil then B = true
      [] tre(K V T1 T2) then
      {Og {BinTre T1} {BinTre T2} B}
      else B = false end
   end
end
```

```
local
   proc {OgSaa BP1 BP2 ?B}
      if {BP1} then B = {BP2} else B = false end
   end
in
   proc {BinTre2 T ?B}
      case T
      of nil then B = true
      [] tre(K V T1 T2) then
         {OgSaa
          proc {$ B1} {BinTre2 T1 B1} end
          proc {$ B2} {BinTre2 T2 B2} end
          B}
      else B = false end
   end
end
```

Oppgave 2 Lat og annen programmering i Oz (4 punkter)

a) Hva er hensikten med lat programmering? Forklar med et eksempel. Vis hvordan et uttrykk med syntakssukkeret `lazy` blir oversatt til `ByNeed` i kjernespråket.

b) Implementer lat `ListMultiplesOf` som genererer alle heltall fra og med 0 som er multipler av et heltallsargument. Hva blir resultatet av å evaluere:

```
{Browse {ListMultiplesOf 7}}
{Browse [A B C] = {ListMultiplesOf 7}}
```

c) Implementer `fun {PickListItem Itemno List}` som plukker element nummer `Itemno`, nummerert fra 1, fra listen `List`.

d) Hva er resultatet av `Browse`-kallene under? Forklar.

```
OI1={ListMultiplesOf 4}
OI2={ListMultiplesOf 8}
{Browse local N in
     {PickListItems N OI1}
  end}
{Browse local H|T=[10 100] in
```

```
     T
     end}
local LowerBound Y C in
   Y = 5
   proc {LowerBound X ?Z}
      if X>=Y then Z=X else Z=Y end
   end
   {LowerBound 1 C}
   {Browse C}
end
{Browse
   local Y in
      local X=[1 2] in
         X.1=Y.1
       end
   end}
local Dodat X = 2 Y in
   proc {Dodat Z} Z = 1 end
   {ByNeed Dodat X}
   {Browse Y}
   Y = X + 1 end
```

Oppgave 3 Programmeringsparadigmer og språkteori (5 punkter)

a) Hvilke deler har en komplett definisjon av et programmeringspsråk? Finnes det forskjellige måter å definere de forskjellige delene? Forklar!

b) I Oz brukes logiske variable. Hva er spesielt med disse? Forklar hvordan de implementeres i kjernemaskinen.

c) Skisser hovedelementene i kjernespråket til Oz (se punkt a)?

d) Forklar hvorfor utvidelsen med porter i Oz ikke kan bruke dataflyt- (logiske) variable. Definer semantikken til `{NewPort Stream Port}` og `{Send Port V}`.

e) Hva er et typesystem i et programmeringsspråk? Beskriv typesystemet i Oz.

Oppgave 4 Grammatikker og parsing (4 punkter)

I denne oppgaven skal du lage en rekursiv nedstigningsparser for en del av spørrespråket SQL definert ved følgende grammatikk:

⟨*select stmt*⟩ ::= 'select' ⟨*name list*⟩ 'from' ⟨*name list*⟩ 'where' ⟨*expr*⟩

⟨*name list*⟩ ::= ⟨*name*⟩ | ⟨*name*⟩ ',' ⟨*name list*⟩

⟨*expr*⟩ ::= ⟨*name*⟩ '=' ⟨*value*⟩

Anta at vi har en leksikalsk analysator (eng. *tokenizer* som leser SELECT-setninger og gjør den om til egnete posttyper i Oz. (F.eks.: `val(V) name(L)` og literaler for nøkkelordene. Ignorer feilretting og feilhåndtering i parseren.

a) Er strukturen i grammatikken egnet for en rekursiv nedstigningsparser? Forklar. Hvis den ikke er egnet, vis ekvivalens-bevarende transformasjoner som må gjøres for å lage en egnet grammatikk.

b) Skriv i Oz en funksjon `{Expr In ?Out}` som kjenner igjen en instans av `<expr>` fra starten av leksem-lista (eng: token list) `In`, for eksempel lista `[name(adams) '='val(4) | R]`. Funksjonen må binde resten av lista til `Out` og returnere et tuppel, for eksempel `equal(adams 4)`, som inneholder informasjonen fra den delen av lista som ble gjenkjent.

c) Skriv en tilsvarende funksjon som kjenner igjen en navneliste i starten av leksemlista og returnerer en liste av tupler som inneholder informasjon om den gjenkjente delen av lista.

d) Fullfør parseren ved å lage en funksjon `{SelectStmt In ?Out}` som kjenner igjen en hel SELECT-setning, og returnerer et tilsvarende komplett syntakstre for setningen.