NTNU,
Institutt for Datateknikk og Informasjonsvitenskap
Faglig kontakt under eksamen:
Ole Edsberg (tlf. 952 81 586)

Eksamen i emne
TDT4165 PROGRAMMERINGSSPRÅK

Fredag 6. august 2004, kl. 0900 – 1300

Oppgaven er kvalitetssikret av:

Per Holager (faglærer) Tor Stålhane (kontrollør)

Hjelpemidler: D (Ingen trykte eller håndskrevne hjelpemidler er tillatt. Godkjent lommekalkulator er tillatt.)

Les hele oppgavesettet før du begynner svare. Svar kort, klart og tydelig:
Er svaret uklart eller lengre enn nødvendig, trekker dette ned.

All programkode er/ skal være i oz, om ikke annet er angitt. Der det står "mater inn" betyr det inn i mozart-systemet, Oz-vinduet. Feilmeldinger kommer i Oz Compiler-vindet, resultater vanligvis i Browser-vinduet. Layout av tekst (innrykk, linjeskift o.l) fra Browser-vinduet er ikke gjengitt nøyaktig i dette oppgavesettet.

Oppgave 1 Flervalg (30%, hver deloppgave teller likt)

Denne oppgaven skal besvares på vedlagte svarark (side 7 i dette oppgavesettet). Sett bare ett kryss for hver deloppgave: Velg det svaret du mener er mest riktig. Hvis du krysser feil, kan krysset slettes ved å fylle hele ruta.

a) Man mater inn linja

```
declare X Y = en(X Y) {Browse Y}
```

Hva blir resultatet?

1: feilmelding "variable X not introduced"
2: feilmelding "variable Y not introduced"
3: uendelig rekursjon, dvs. ingen reaksjon
4: "en(X Y)"
5: "en(_ _)"
6: "en(X en(X en(X en(X en(X en(X en(X en(X en(X en(X en(X
en(X en(X en(,,, ,,,)))))))))))))))"

b) Man mater inn linja

declare E = 4711 {Browse E} declare E = 13

Hva blir resultatet?

1: feilmelding "variable E already declared"
2: først "4711" som så raskt erstattes av "13"
3: "4711"
4: "13"
5: "4711" og "13" på hver sin linje

c) Man mater inn linja

declare S T = s(_ S) U = s(S s) {Browse T} T = U

Hva blir resultatet?

1: feilmelding "expression at statement position" (posisjon for "_"-tegnet)
2: feilmelding "equality constraint failed"
3: "s(s s)"
4: "s(_ S)"
5: "s(s s(s s(s s(s s(s s(s s(s s(s s(s s(s s(s s(s s(s s(s s(,,, ,,,))))))))))))))))"

d) Man mater inn linja

declare K L M = nil | L | K L = nil | K K = nil

Hvilken verdi får M:

1: [nil nil]
2: [nil [nil]]
3: [nil [nil] nil]
4: [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil [nil ,,,]]]]]]]]]]]]]]]]
5: ingen av disse

e) Man mater inn

```
declare fun {G H I}
   case H
   of nil then I
   [] [nil] | J  then {G  J  I+1}
   else {G  H.2  I}
   end
end
{Browse {G  [nil|nil nil|nil nil nil|nil]  0}}
```

Hva blir resultatet?

1: feilmelding "expecting []"
2: "1"
3: "3"
4: "4"
5: "7"

f) Man mater inn

```
declare fun {M N P Q}
   case P
   of (R|S) | T then {M  N  T  {M  N  R|S  Q}}
   [] R|S then {M  N  S  {N  R  Q}}
   else Q
   end
end
fun {T U V}  V|U  end
{Browse {M  T  [a  [b c d] [[e f]]]  1}}
```

Hva blir resultatet?

1: "[a b c d e f]"
2: "[a b c d e f 1]"
3: "[[[[[[1 a] b] c] d] e] f]
4: "(((((1|a)|b)|c)|d)|e)|f
5: "(1|a|(b|c|d)|((e|f)))

g) Kan man bruke BNF for å spesifisere (deler av) hva en leksikal-analysator skal gjøre?

1: ja, vanligvis
2: ja, men bare hvis man har en kontekst-avhengig grammatikk
3: nei

h) Gitt en grammatikk i EBNF:

<tab def> ::= 'create table' {<col def>}$^{+}$
<col def> ::= 'name' <data type> [<constraint>]
<data type> ::= 'varchar(number)' | 'integer'
<constraint > ::= 'not null' ['unique']

Her er startsymbolet <tab def>, tekst i et par enkeltfnutter er et leksikalsymbol / atom.
Hvilken av de følgende er lovlige stringer i denne grammatikken?

1: create table
2: create table colA varchar(5)
3: create table colA varchar(5) unique colB integer
4: create table colA varchar(5) not null primary key colB integer unique
5: create table name varchar(number) not null unique name integer not null unique

i) Hva er den viktigste fordelen av statisk typesjekking?

1: slurvefeil resulterer ofte i feilmelding
2: ineffektiv kode vil ofte resultere i feilmelding
3: programmet blir vesentlig kortere
4: kompilert kode blir mer kompakt
5: programmet kan lettere kjøre på små, bærbare cpu-er

j) Studer følgende kode:

```
declare
X={NewCell  0}
fun {X2}
    E= @X in E*E
end
fun {Sum  A  B  F}
    if A =< B then
        X := A
        {F}+{Sum  A+1  B  F}
    else 0
    end
end

declare A={Sum  0  4  X2} {Browse A}
```

Hvilken metode for parameteroverføring simuleres her (parameter F)?

1: *Kall-ved-referanse* (engelsk *call by reference*)
2: *Kall-ved-variabel* (engelsk *call by variable*)
3: *Kall-ved-verdi* (engelsk *call by value*)
4: *Kall-ved-verdi/resultat* (engelsk *call by value-result*)
5: *Kall-ved-navn* (engelsk *call by name*)
6: *Kall-ved-behov* (engelsk *call by need*)

k) Gitt de vanlige erklæringene fra boka eller biblioteket for SolveAll etc. og en rettet graf ved

```
declare fun {Kant}
   choice kant(1 2) [] kant(2 1) [] kant(2 3) [] kant(1 4) [] kant(2 4) [] kant(4 1) end
end
```

Vi skal så utføre

```
declare fun {Finn}  ...  end
{Browse {SolveAll Finn}}
```

Hvilken av de følgende fullstendige utgavene av Finn vil føre til at alle løkker av lengde to, altså par av noder som har kanter til hverandre, blir skrevet ut?

1: fun {Finn} kant(A B)=kant(B A) in løkke(A B) end
2: fun {Finn} kant(A={Kant} kant(B={Kant} A)) in (A B) end
3: fun {Finn} løkke(kant(A B)={Kant} kant(B A)={Kant}) end
4: fun {Finn} kant(A B)={Kant} kant(B A)={Kant} in løkke(A B) end
5: fun {Finn} B={Kant kant(A {Kant kant(B A)})} in løkke(A B) end

l) Hvilken av de følgende kodesekvensene vil gå i vranglås (engelsk *Deadlock*)?

1: declare A=1 B C D E thread C=A+1 B=D+1 end D=C+1 E=B+1
2: declare A=1 B C D E thread C=A+1 B=C+1 end D=B+1 E=C+1
3: declare A=1 B C D E thread C=A+1 B=A+1 end D=B+1 E=C+1
4: declare A=1 B C D E thread C=A+1 B=E+1 end D=C+1 E=B+1
5: declare A=1 B C D E thread C=A+1 B=D+1 end D=A+1 E=B+1

Oppgave 2 Spørsmål (20%, hver deloppgave teller likt)

Skriv ikke mer enn en halv side i normal skriftstørrelse som svar på hvert av del-spørsmålene. Kodeeksempler trenger ikke være fullstendige, bare delene som er viktige for å forstå hva du mener trenger være med.

a) Forklar begrepet *utbyttbarhet* (engelsk s*ubstitution property*) i forbindelse med objekt-orientert programmering. Gi et kort eksempel av kode som ikke har denne egenskapen.

b) Forklar begrepet *ikke-determinisme* (engelsk *nondeterminism*). Gi et kort eksempel (uten bruk av Browse, Show eller lignende) av kode hvor slik oppførslel blir synlig.

c) Forklar hva som gjøres ved *unifisering*. Gi et kort eksempel.

d) Forklar forskjellen på *regulære* grammatikker, *kontekst-fri* grammatikker og *kontekst-avhengige* grammatikker. Gi et veldig kort eksempel på en kontekst-fri grammatikk.

Oppgave 3 Høyere ordens programmering (20%; deloppgave a:8%, b:8%, c:4%)

a) Skriv en funksjon FromDec som konverterer ei liste av desimalsiffer til en vanlig intern binær integer. For enkelthets skyld kan FromDec ha en akkumulator som skal ha verdien 0 i det ytterste kallet, for eksempel {FromDec [4 7 1 1] 0} = 4711.

b) Denne funksjonen skal generaliseres slik at den kan konvertere for andre tallsystemer. Skriv en funksjon MakeFrom som tar et basetallet som parameter og gir som resultat en funksjon som konverterer ei liste siffer i tallsystemet med det basetallet til en vanlig intern binær integer. Man skal for eksempel ha at {{MakeFrom 2} [1 0 1 1] 0} = 11 og at {{MakeFrom 16} [6 4] 0} = 100 .

c) Skriv en funksjon FromHex som tar som parameter ei liste hexadesimale siffer, og gir som resultat en vanlig intern binær integer. Bruk funksjonen fra b) for å redusere skrivearbeidet. Man skal for eksempel ha {FromHex [15 15 0 0]} = 65280.

Oppgave 4 Grammatikker og parsing (30%; deloppgave a:3%, b:6%, c:9%, d:12%)

I denne deloppgaven skal du skrive en rekursiv-nedstignings-parser for en sterkt forenklet utgave av grammatikken for betingelser i SQL:

<condition> ::= <bool term> | <condition> 'OR' <bool term>
<bool term> ::= <bool factor> | <bool term> 'AND' <bool factor>
<bool factor> ::= <value> <comp op> <value> | '(' <condition> ')'
<comp op> ::= '<' | '<=' | '>' | '>=' | '=' | '<>'
<value> ::= <name> | <int const>

Her skal tekst i enkelt-apostrofer , <name> og <int const> håndteres som leksikal-symboler dvs. atomer. Hvis innputt til leksikal-analysatoren for eksempel er

err>0 AND err<1000 OR sum<0

kan innputt til parseren være

[name(“err”) '>' intConst(0) 'AND' name(“err”) '<' intConst(1000)
'OR' name(“sum”) '<' intConst(0)]

Ikke legg arbeid i å håndtere syntaks-feil i inn-dataene: standard feilreaksjon i mozart-systemet er godt nok.

a) Tegn parse-tre og syntaks-tre for eksemplet

err>0 AND err<1000 OR sum<0

Velg representasjonen slik at syntax-treet blir lite og enkelt.

b) Er strukturen i grammatikken egnet for en rekursiv-nedstignings-parser? Forklar kort. Hvis den ikke er egnet, vis en ekvivalent grammatikk som er egnet. Forklar kort hvilke transformasjoner du bruker.

c) Skriv en funksjon BoolFactor som gjenkjenner en forekomst av <bool factor>: Den skal som
* inn-parameter ta ei liste leksikal-symboler som starter med symbolene for en <bool factor>,
* gi som ut-parameter resten av inn-lista, etter symbolene som hørte til <bool factor>-en.

Skriv også en tilsvarende funksjon Value som gjenkjenner <value>. For begge skal funksjonsresultatet skal være egnet til å representere den gjenkjente delen i et syntaks-tre. Du kan kalle en funksjon som gjenkenner <condition> uten å definere denne (det skal du gjøre i neste deloppgave).

d) Skriv tilsvarende funksjoner Condition som gjenkjenner <condition> og BoolTerm som gjenkjenner <bool term>. Bruk BoolFactor fra forrige deloppgave der den trengs, innfør eventuelle hjelpefunksjoner som må til, slik at du har en fullstendig parser for grammatikken over, evt. den modifiserte du lagde i deloppgave b).

NTNU, IDI,
6.august 2004

Studentnummer:_____

Svarark for Oppgave 1:

flervalgsdelen av Eksamen i
TDT4165 Programmeringsspråk,

svar-alternativ nr.:	1	2	3	4	5	6
deloppgave						
a)	ڤ	ڤ	ڤ	ڤ	ڤ	ڤ
b)	ڤ	ڤ	ڤ	ڤ	ڤ	
c)	ڤ	ڤ	ڤ	ڤ	ڤ	
d)	ڤ	ڤ	ڤ	ڤ	ڤ	
e)	ڤ	ڤ	ڤ	ڤ	ڤ	
f)	ڤ	ڤ	ڤ	ڤ	ڤ	
g)	ڤ	ڤ	ڤ			
h)	ڤ	ڤ	ڤ	ڤ	ڤ	
i)	ڤ	ڤ	ڤ	ڤ	ڤ	
j)	ڤ	ڤ	ڤ	ڤ	ڤ	ڤ
k)	ڤ	ڤ	ڤ	ڤ	ڤ	
l)	ڤ	ڤ	ڤ	ڤ	ڤ	