UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i	IN 211 — Programmeringsspråk
Eksamensdag:	4. desember 2002
Tid for eksamen:	9.00 – 15.00
Oppgavesettet er på 10 sider.	
Vedlegg:	Ingen
Tillatte hjelpemidler:	Alle trykte og skrevne

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

Innhold

1	**Kjøresystemer** (vekt 20%)	side 2
2	**ML-map** (vekt 25%)	side 3
	Hypervektorer: Innledning	side 6
3	**Hypervektorer: Grammatikk** (vekt 25%)	side 7
4	**Hypervektorer: Prolog** (vekt 15%)	side 8
5	**Hypervektorer: ML** (vekt 10%)	side 10
6	**Hypervektorer: Språkvalg** (vekt 5%)	side 10

Først noen generelle råd og bemerkninger:

- Oppgavesettet består av seks uavhengige deler, men oppgavene 3–6 forutsetter alle at du har lest innledningen på side 6.
- Prosentene angitt på hver oppgave antyder hvor mye vekt de forskjellige delene vil bli tillagt ved sensuren. De enkelte deloppgavene kan bli gitt ulik vekt avhengig av vanskelighetsgrad.
- Om oppgaveteksten på noe punkt er uklar eller upresis, kan du gjøre egne presiseringer. Formulér i så fall disse tydelig i oppgavebesvarelsen din.
- Legg vekt på å gi korte og klare forklaringer.
- Les oppgavene nøye!

Lykke til!

Ragnhild Kobro Runde og Gerhard Skagestein

(Fortsettes på side 2.)

Figur 1: Eksempel-program

```
void main() {
    int x = 1;
    int y = 2;

    void C() {
        print(y);
    }

    void A() {

        void B() {
            int y = 3;
            if (x > 0) {
                x--;
                B();
            } else {
                y = 4;
                C();
            }
        }

        void D() {
            int x = -y;
            B();
        }

        D();
    } // end A

    A();
}
```

Oppgave 1 Kjøresystemer (vekt 20%)

1a Statisk skop

I figur 1 er det vist et lite program skrevet i C4". Med statiske skopregler, hvilken verdi blir skrevet ut av print-setningen?

Tegn runtime-stakken (med spesiell vekt på variable, statisk og dynamisk link) slik den ser ut rett før print-setningen utføres.

(Fortsettes på side 3.)

1b Dynamisk skop

1. Med dynamisk skop, vil det fortsatt være nødvendig å bruke statisk og dynamisk link?
2. Se igjen på programmet i figur 1 på forrige side. Hva ville blitt skrevet ut hvis vi istedenfor hadde hatt dynamiske skopregler? Tegn runtime-stakken slik den nå ville sett ut rett før utførelsen av print-setningen.

1c Parameteroverføring

En variant av call-by-name (name-overføring) er call-by-text. I begge tilfellene evalueres argumentene først ved bruk, forskjellen er at de ved call-by-text evalueres i forhold til omgivelsen til den kalte metoden (og ikke omgivelsen til kalleren, slik som ved name-overføring).

1. Vis (ved eksempel) at call-by-text kan gi et annet resultat enn call-by-name.
2. Hva mener du må gjøres av kompilatoren og kjøresystemet for å få til call-by-text? Skriv kort.

Oppgave 2 ML-map (vekt 25%)

Et map er en abstrakt datatype for å mappe nøkler (*keys*) til verdier (*values*). Et map kan ikke inneholde like nøkler, og hver nøkkel mapper til maksimalt en verdi.

I ML kan et generelt map angis ved følgende signatur:

```
signature Map_def =
sig
  type (''key, ''val) Map
  exception noValue
  val empty        : (''key, ''val) Map
  val size         : (''key, ''val) Map -> int
  val containsKey  : (''key, ''val) Map * ''key -> bool
  val containsValue: (''key, ''val) Map * ''val -> bool
  val get          : (''key, ''val) Map * ''key -> ''val
  val put          : (''key, ''val) Map * ''key * ''val
                     -> (''key, ''val) Map
  val remove       : (''key, ''val) Map * ''key
                     -> (''key, ''val) Map
  val union        : (''key, ''val) Map * (''key, ''val) Map
                     -> (''key, ''val) Map
end;
```

(Fortsettes på side 4.)

Her skal `empty` representere et tomt map (uten nøkler), og funksjonene være som følger:

- `size(m)`: Returnerer antall (nøkkel,verdi)-mappinger i `m`.
- `containsKey(m,k)`: Returnerer `true` hvis `m` inneholder en mapping for nøkkelen `k`.
- `containsValue(m,v)`: Returnerer `true` hvis det finnes minst en nøkkel som `m` mapper til verdien `v`.
- `get(m,k)`: Returnerer den verdien som nøkkelen `k` mappes til i `m`. Hvis `k` ikke har noen verdi i `m`, gis unntaket `noValue`.
- `put(m,k,v)`: Assosierer verdien `v` med nøkkelen `k` i `m`. Hvis `m` allerede inneholder en mapping for `k`, vil altså den gamle verdien bli erstattet.
- `remove(m,k)`: Fjerner mappingen for nøkkelen `k` i `m`, hvis den finnes.
- `union(m1,m2)`: Slår sammen to map. Hvis samme nøkkel finnes i både `m1` og `m2`, beholdes mappingen fra `m2`.

2a Liste-implementasjon av map

Gi en komplett implementasjon av denne signaturen slik at et map implementeres som en liste av (nøkkel, verdi)-par, det vil si:

```
type (''key, ''val) Map = (''key * ''val) list;
```

2b Bruk av map

Gitt et map som for hver person (string) som nøkkel gir en liste av personer (string list) som denne har besøkt det siste året, for eksempel:

Nøkkel		**Verdi**
`"anne"`	~	`["liv", "ole"]`
`"liv"`	~	`["ole", "anne", "lise"]`
`"ole"`	~	`["lise"]`

Lag en funksjon `antallbesokt` som tar et slikt map, og returnerer et map som for hver person gir antall personer denne har fått besøk av. For eksempelet gitt over, skal altså funksjonen returnere et map bestående av mappingene:

Nøkkel		**Verdi**
`"anne"`	~	1
`"lise"`	~	2
`"liv"`	~	1
`"ole"`	~	2

(Fortsettes på side 5.)

Funksjonen `antallbesokt` skal altså ta et map av typen `(string, string list) Map`, og returnere et map av typen `(string, int) Map`.

For å aksessere et map, skal du i denne oppgaven *ikke* bruke noen kunnskaper om hvordan det er implementert. Et map kan istedenfor aksesseres via funksjonene i signaturen. I tillegg til funksjonene angitt nederst på side 3, kan du anta at denne inneholder de to vanlige map-funksjonene:

- `keys(m)`: Returnerer en liste med alle nøklene i `m`.
- `values(m)`: Returnerer en liste med alle verdiene i `m`.

Disse trenger du altså ikke å implementere.

2c Map implementert som funksjonsrom

Som en alternativ implementasjon av et map, kan vi tenke oss funksjonsrommet fra nøkler til verdier, det vil si:

```
type (''key, ''val) Map = ''key -> ''val;
```

- Med denne typen vil det imidlertid ikke være mulig å implementere alle elementene i signaturen på side 3. Angi hva som ikke kan implementeres og hvorfor.
- Implementer de delene av signaturen som er mulige med det gitte funksjonsrommet.

(Slutt oppgave 2.)

(Fortsettes på side 6.)

Hypervektorer: Innledning

Resten av dette oppgavesettet vil handle om hypervektorer i ulike varianter.

En vektor er en liste med tall, for eksempel $\mathbf{a} = (a_1, a_2, \ldots a_n)$.

Skalarproduktet (også kalt prikkproduktet, "dot product") mellom to vektorer med like mange (n) elementer (a_1, a_2, ..., a_n) og (b_1, b_2, ..., b_n) er definert som $\Sigma(a_i * b_i)$, altså summen av produktene av tilsvarende elementer i de to vektorene. Skalarproduktet symboliseres med operatoren $\bullet$.

Eksempel:

$$\begin{aligned}&(1,2,3,4,5) \bullet (10,20,30,40,50)\\ &= (1*10 + 2*20 + 3*30 + 4*40 + 5*50)\\ &= 550\end{aligned}$$

Operatoren $\bullet$ kan generaliseres til å gjelde også mellom såkalte *hypervektorer*, som er vektorer som har vektorer som elementer. Vi definerer skalarproduktet mellom to hypervektorer som $\Sigma(a_i \bullet b_i)$, altså summen av skalarproduktene av tilsvarende elementer i de to vektorene.

Eksempel:

$$\begin{aligned}&((1,2),(3,4,5)) \bullet ((10,20),(30,40,50))\\ &= ((1,2) \bullet (10,20) + (3,4,5) \bullet (30,40,50))\\ &= ((1*10 + 2*20) + (3*30 + 4*40 + 5*50))\\ &= (50 + 500)\\ &= 550\end{aligned}$$

Vi observerer at operatoren $\bullet$ er rekursiv, og degenererer til en operasjon med en vanlig multiplikasjon når operandene er tall (skalarer).

Hovedanvendelsen av dette prinsippet er å spare lagerplass og forenkle beregningene i de tilfellene der mange av undervektorene er (). Tomme vektorer skal tolkes som en vektor av passende lengde, der alle elementene er enten 0 eller tomme vektorer. For eksempel vil skalarproduktet av ((1,2),(3,4)) og ((5,6),()) være det samme som skalarproduktet av ((1,2),(3,4)) og ((5,6),(0,0)), det vil si 17.

I programmeringssammenheng er det naturlig å representere en vektor ved hjelp av en liste. En hypervektor vil dermed være representert som en liste av lister.

(Fortsettes på side 7.)

Oppgave 3 Hypervektorer: Grammatikk (vekt 25%)

For å kunne håndtere hypervektorer kan vi lage oss et enkelt høynivåspråk. Her følger et eksempel på et program i dette språket. (For at språket skal bli praktisk brukbart, må vi også ha en print-funksjon, men det ser vi bort fra her.)

$$
\begin{aligned}
&X1 \leftarrow [1,2];\\
&Y1 \leftarrow [3,4,5];\\
&Ti \leftarrow 10;\\
&R \leftarrow [X1, Y1] \bullet [[10,\ 2 \bullet Ti], [3 \bullet Ti\ , 40, 50]];
\end{aligned}
$$

Her er en grammatikk i utvidet BNF for dette språket:

$$
\begin{aligned}
\langle prog \rangle &\rightarrow \langle assign \rangle^{+}\\
\langle assign \rangle &\rightarrow \texttt{var} \leftarrow \langle expr \rangle \texttt{;}\\
\langle expr \rangle &\rightarrow [\![\texttt{var} \mid \texttt{[}\ \langle expr \rangle\ [\![\texttt{,} \langle expr \rangle]\!]^{*}\ \texttt{]} \mid \texttt{tall}]\!]\ [\![\bullet\ \langle expr \rangle]\!]^{?}
\end{aligned}
$$

var og **tall** kan her betraktes som grunnsymboler, på lik linje med $\bullet$, $\leftarrow$, **[**, **]**, **;** og **,**

Legg merke til at språket ikke omfatter noen spesiell operator for multiplikasjon av tall (skalarer). Merk også at grammatikken tillater uttrykk som for eksempel $[3,5] \bullet 7$, selv om vi ikke betrakter dette som semantisk korrekt.

3a Typing

1. Hva slags typekompatibilitetssystem har dette språket?
2. Operatoren $\bullet$ er polymorf, men hva slags polymorfisme er det snakk om?
3. Er språket sterkt typet? Begrunn svaret.

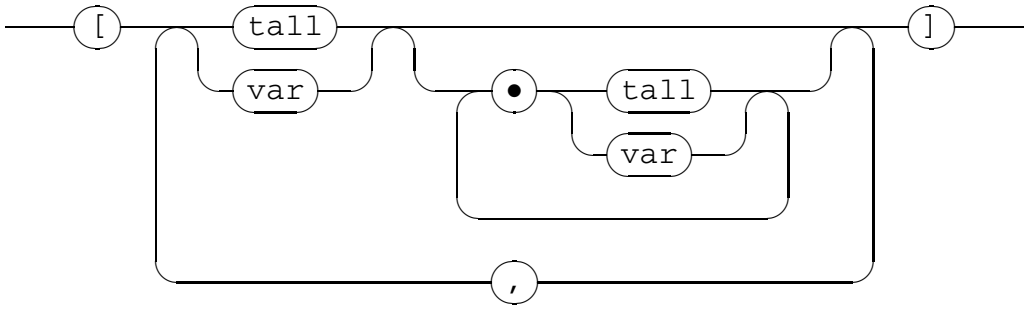
3b LL(1)

1. Skriv om grammatikken ovenfor til en grammatikk som er formulert i standard BNF og som tilfredsstiller kravene til LL(1).
2. Vis at ditt forslag til grammatikk tilfredsstiller kravene til LL(1).

(Fortsettes på side 8.)

3c Regulære uttrykk

Grammatikken gitt på side 7 kan ikke gjøres regulær, da vi kan ha vilkårlig dyp nesting av parenteser. Vi kan imidlertid lage et forenklet språk der vi bare har ett nivå med parenteser. Dette kan beskrives ved et syntaksdiagram (jernbanediagram) som følger:



Vi antar at `tall` og `var` er parsert allerede, slik at de her kan tolkes som grunnsymboler.

1. Skriv en regulær grammatikk, eventuelt et regulært uttrykk, for dette språket.
2. Konstruer en ikke-deterministisk automat ut fra syntaksdiagrammet.
3. Gjør automaten fra forrige punkt om til en deterministisk automat.

Oppgave 4 Hypervektorer: Prolog (vekt 15%)

Følgende Prolog-program beregner skalarproduktet av to vektorer med tall som elementer:

```
dotProd(_,[],0).
dotProd([],_,0).
dotProd([H1|T1],[H2|T2],Result) :- dotProd(T1,T2,R),
                                   Result is R + (H1*H2).
```

På neste side følger en trace-utskrift av utførelsen av programmet for spørringen `dotProd([1,2],[10,20],R).`

(Fortsettes på side 9.)

```
| ?- dotProd([1,2],[10,20],R).
        1      1 Call: dotProd([1,2],[10,20],_226) ?
        2      2 Call: dotProd([2],[20],_753) ?
        3      3 Call: dotProd([],[],_1171) ?
?       3      3 Exit: dotProd([],[],0) ?
        4      3 Call: _753 is 0+2*20 ?
        4      3 Exit: 40 is 0+2*20 ?
?       2      2 Exit: dotProd([2],[20],40) ?
        5      2 Call: _226 is 40+1*10 ?
        5      2 Exit: 50 is 40+1*10 ?
?       1      1 Exit: dotProd([1,2],[10,20],50) ?

R = 50 ? ;
```

4a Spørringer

1. Tegn opp det komplette søketreet for denne spørringen.
2. Hvor mange flere løsninger finnes for denne spørringen, og hvilke verdier vil da bli bundet til R?
3. Lag to versjoner av dotProd som ikke produserer flere løsninger. Den ene løsningen skal baseres på bruk av cut, den andre løsningen skal ikke gjøre bruk av noen av de prosedurelle aspektene ved Prolog.
4. Som kjent kan vi i mange Prolog-programmer velge nokså fritt hvilke parametre som skal være input, og hvilke som skal være variable som bindes til verdier gjennom utførelsen. Vil en spørring som eksempelvis dotProd([1,2],X,50) gi et fornuftig resultat? Begrunn svaret.

4b Hypervektorer

Utvid Prolog-programmet slik at det kan håndtere hypervektorberegninger. For eksempel skal spørringen
`hyperDotProd([[1,2],[3,4,5]],[[10,20],[30,40,50]],R).`
gi `R = 550` som svar.

Husk at vi generelt kan ha vilkårlig dyp nesting av vektorer.

Hint: Det innebygde predikatet `number(X)` kan brukes for å teste om `X` er et tall.

(Fortsettes på side 10.)

Oppgave 5 Hypervektorer: ML (vekt 10%)

En hypervektor kan i ML representeres som en liste der elementene i listen enten alle er tall eller alle selv er (hyper-)vektorer.

1. Vi definerer en hypervektor ved

   ```
   type hypervektor = hyperelement list;
   ```

   der et hyperelement er enten et tall eller en liste (av hyperelementer).

   Definer datatypen `hyperelement`.

2. Hvordan vil hypervektorene `((1,2),(3,4,5))` og `((10,20),(30,40,50))` se ut med denne representasjonen?

3. Skriv en funksjon

   ```
   fun hyperprod(v1:hypervektor,
                 v2:hypervektor):int = ...;
   ```

   som beregner hyperproduktet av `v1` og `v2`. For eksempel skal hyperproduktet av de to hypervektorene fra forrige punkt gi 550 som svar.

   Husk at vi generelt kan ha vilkårlig dyp nesting av vektorer.

4. Istedenfor å bruke en egendefinert type for å representere hypervektorer, kan det være fristende å la parametrene `v1` og `v2` til `hyperprod` være av typen `'a list`. Vil dette være mulig her? Begrunn kort.

Oppgave 6 Hypervektorer: Språkvalg (vekt 5%)

Anta at du skal skrive et program for å evaluere hypervektoruttrykk som for eksempel ((1,2),(3,4,5)) • ((10,20),(30,40,50)). Gi inntil fem kriterier som du vil legge vekt på ved valget av programmeringsspråk.

Ut fra disse kriteriene, ville du valgt et imperativt, funksjonelt eller logisk programmeringsspråk? Begrunn svaret.