Prolog Tentamen: 1 juni 1999

vraag	*aantalpunten*
1.Unificaties	15
2.Geven van afleidingsboom	15
3.Cut operator	20
4.Programma schrijven	15
5.Grammatica	25

Cijfer = (aantal punten + 10) /10

Veel succes!

Unificaties (15 punten)

Wat is het antwoord van de Prolog interpreter op de onderstaande queries?

```
 1. ?- [X,Y,Z] = [john,likes,fish].
 2. ?- [cat] = [X|Y].
 3. ?- [X,Y|Z]=[mary,likes,wine].
 4. ?- [[the,Y]|Z] = [[X,hare],[is,here]].
 5. ?- [X,Y|Z,W] = [1,2,3,4].
 6. ?- [golden|T]=[golden,norfolk].
 7. ?- [vale,horse]=[horse,X].
 8. ?- [white|Q]=[P|horse].

 9. ?- a(b,C,d(e,F,g(h,i,J)))=a(F,c,d(J,F,g(h,i,J))).
10. ?- f(1,2)=X(Y,Z).

11. ?- Y = 3, X is 3 + Y.
12. ?- X is 3 + Y, Y = 3.
13. ?- X is 12, X =:= 6 + 6.
14. ?- X = 12, X = 6 + 6.
15. ?- f(A,B) == f(X,Y).
```

Vraag 2: Afleidingsboom (15 punten)

Gegeven de volgende delete procedure:

```
delete([X|Xs],X,Ys) :- delete(Xs,X,Ys).
delete([X|Xs],Z,[X|Ys]) :-  not X = Z, delete(Xs,Z,Ys).
delete([],X,[]).
```

Geef de complete afleidingsboom van de query delete([1,2,4,2],2,A).

Vraag 3: Cut operator (20 punten)

3a: if-then-else (15 punten)

Het gedrag van *if-then-else(P,Q,R)* is zoals je zou verwachten. In een procedurele lezing is dit namelijk als volgt:

als P waar is dan moet je Q doen, en anders moet je R doen.

1. Schrijf twee versies van de if-then-else procedure. Een versie waarin je geen cut operator gebruikt en een versie waarin je wel een cut operator(en) gebruikt.
2. Geef bij de cut-versie aan wat de functie van iedere cut operator is.
3. Motiveer van iedere cut-operator in de cut-versie van je *if-then-else* procedure of het een rode of een groene cut is.

3b: (10 punten)

Gegeven het volgende programma:

```
foo(1).
foo(2).
foo(3).
```

Wat is het antwoord van de Prolog interpreter (met backtracken) op de volgende query:

```
?- foo(X), !, foo(Y).
```

Vraag 4: Programma (15 punten)

"maplist" is een ingebouwd predicaat. Het heeft de volgende definitie:

```
maplist(+Pred, ?List1, ?List2)
   Pas het predicaat Pred toe op alle opeenvolgende paren uit
   List1 en List2.
   Faalt als Pred niet toegepast kan worden op een paar.
```

Een voorbeeld van het toepassen van maplist:

```
?- maplist(kwadraat, [0, 1, 2], X).

X = [0, 1, 4]
```

Waarbij de definitie voor kwadraat is:

```
kwadraat(X,Y) :- Y is X*X.
```

Schrijf de code voor de maplist procedure.

Vraag 5: Definite clause grammars (DCG's) (25 punten)

Hieronder vind je met behulp van DCG's een beschrijving van een pad in een doolhof. In het doolhof kun je alleen stappen vooruit (v) , achteruit (a), linksaf (l) en rechtsaf (r) maken. Echter je kunt alleen rechtsaf nadat je vorige stap vooruit was. Verder moet je altijd een veelvoud van 2 (2,4,..) naar rechts gaan voordat je weer een andere richting uit kunt.

```
doolhofpad --> stap.
doolhofpad --> stap,doolhofpad.
stap --> [v].
stap --> [l].
stap --> [a].
stap --> [v], staprechts.
staprechts --> [r],[r].
staprechts --> [r],[r], staprechts.
```

Voorbeelden van een goed pad zijn: "v v l a v r r a" en "v r r r r l".

In Prolog:

```
?- doolhofpad([v,v,l,a,v,r,r,a],[]).
Yes
?- doolhofpad([v,r,r,r,r,l],[]).
Yes
```

Voorbeelden van een fout pad zijn: "v l r r" en "v r r r a".

In Prolog:

```
?- doolhofpad([v,l,r,r],[]).
No
?- doolhofpad([v,r,r,r,a],[]).
No
```

5a: Semantiek van een zin (15 punten)

De semantiek van een zin (bijv. van "v r r a") kun je bepalen door twee stappen. De eerste stap is het opbouwen van een parse-tree tijdens het parseren van de zin. Je geeft de parse-tree dan als argument terug (dus *doolhofpad(-ParseTree,[v,v,l,a,v,r,r,a],[]).*). De tweede stap is het interpreteren van de parse-tree. Neem als semantiek van het pad de eindpositie waar je uiteindelijk terecht komt. Begin te tellen vanaf coordinaat (0,0). Een stap naar rechts telt bij de x-coordinaat 1 op, een stap naar links trekt bij de x-coordinaat 1 af, een stap naar voren telt bij de y-coordinaat 1 op, een stap naar achter trekt bij de y-coordinaat er 1 af. De betekenis van de zin "v r r a" is dus (0,2).

1. Neem de bovenstaande DCG's als uitgangspunt en verander de DCG's zodanig dat de parse-tree als argument teruggegeven wordt. (Dit is dus stap 1.)

2. Wat antwoord de Prolog interpreter nu op:

```
?- doolhofpad(ParseTree,[v,r,r,a],[]).
```

3. Schrijf een procedure "meaning" die de parse-tree als input krijgt en de betekenis (=de coordinaten van het eindpunt) van de parse-tree teruggeeft. (dus: meaning(+Parstree,-Meaning)) (dit is dus stap 2).

4. Wat antwoord de Prolog interpreter nu op:

```
?- doolhofpad(ParseTree,[v,v,l],[]),meaning(ParseTree,Meaning).
```

5b: Semantiek van een zin (10 punten)

Zoals je weet kun je ook in de DCG's Prolog clauses tussen accoladen opnemen. Het wordt daardoor mogelijk om de twee stappen (parse-tree opbouwen en betekenis bepalen van de parse-tree) in 1 stap te doen.

Neem de gegeven DCG's weer als uitgangspunt en verander de DCG's zodat de betekenis in 1 stap bepaald kan worden. De betekenins van de zin moet je dus als extra argument teruggeven.

Voorbeeld:

```
        ?- doolhofpad(Meaning,[v,r,r,a],[]).
Meaning = (0,2)
```