Fakulteten för ekonomi, kommunikation och IT

Logik för dataloger

Laboration 1

Datum: 2007-03-07
Namn: Henrik Bäck
Mathias Andersson
Kurs: MATB14

Innehållsförteckning

Uppgift 1

Exercise 1.3

```
father(kent,anna).
father(kent,rolf).
father(ralf,kent).
mother(rut,anna).
mother(rut,rolf).
male(kent).
male(rolf).
male(ralf).
female(anna).
female(rut).
parent(X,Y):-mother(X,Y);father(X,Y).
diff(X,Y):-X\=Y.

is_mother(X):-mother(X,_).


is_father(X):-father(X,_).

is_son(X):-male(X),(mother(_,X);father(_,X)).

sister_of(X,Y):-
diff(X,Y),female(X),mother(P,X),father(Q,X),mother(P,Y),father(Q,Y).

grandpa_of(X,Y):-father(X,P),father(P,Y).

sibling(X,Y):-
diff(X,Y),((mother(P,X),mother(P,Y));(father(Q,X),father(Q,Y))).
```

Exercise 1.4

En person kan vara sin egen syster eftersom i programkoden kan X,Y kan anta samma värde. När detta sker och X är kvinna kan Y vara samma värde som X eftersom inget förhindrar detta. Det enda villkoret som finns är att Y skall ha en gemensam mor eller far med X. Detta kommer att uppfyllas mycket enkelt i de fall när Y = X.

Det går att lösa detta genom att lägga till en kontroll för detta på ett enkelt sätt och det har redan gjorts i vår föregående uppgift. Koden blir som följer:

```
/* X is sister to Y */
sister_of(X,Y):-
diff(X,Y),female(X),mother(P,X),father(Q,X),mother(P,Y),father(Q,Y).
```

Uppgift 2

Programkod

```
/* sum(+Number1, +Number2, -Summa) */
sum(N1,N2,S):-S is N1 + N2.

/* product(+Number1, +Number2, -Product) */
product(N1,N2,P):-P is N1*N2.

/* factorial(+Number, -Fakultet) */
factorial(0,1).
factorial(N,F):-Q is N - 1, factorial(Q,G), F is G * N.

/* smaller_than_all(+Elem, +List) */
smaller_than_all(_,[]).
smaller_than_all(E, [X|T]):- E<X, smaller_than_all(E,T).

/* exchange_all(+Elm, +Elm2, +List, -NewList) */
exchange_all(_,_,[],[]).
exchange_all(E,H,[E|T],[H|S]):-exchange_all(E,H,T,S).
exchange_all(E,H,[X|T],[X|S]):-X\=E,exchange_all(E,H,T,S).

/* exchange_nth(+N, +Elm, +List, -NewList) */
exchange_nth(_,_,[],_).
exchange_nth(0,E,[_|T],[E|T]).
exchange_nth(N,E,[X|T],[X|S]):- N\=0 ,P is N-1,
exchange_nth(P,E,T,S).

/* subset(+List1, +List2) */
is_member(E,[E|_]).
is_member(E,[_|S]):-is_member(E,S).
subset([],_).
subset([E|T],L):-is_member(E,L),subset(T,L).
```