Hertentamen Logisch Programmeren (LIX003B05)
24 March 2021, 9–12 pm

1. **Yes**, it's true. You may use Learn Prolog Now, your notes, and even SWI to test your predicate definitions.
2. And **no**, you are not permitted to ask help of others or browse the internet for answers.
3. Submit your answers **via Nestor** either as PDF or as pictures taken with your camera.
4. For questions about the exam, you can ask me via **johan.bos@rug.nl** until 5pm.
5. You have an additional 10 minutes to scan, prepare and submit the exam via Nestor.
6. Before you start, however, please fill in, sign, and submit the **declaration** via Nestor (it is in the same Nestor folder as the exam).

Question 1. **The first hurdle** (20 points)

All of the following unification attempts fail – explain why. Give a short explanation for each case.

```
(a) ?- bee = bea.
(b) ?- bea = bea(cee).
(c) ?- bea(0) = bea(0,0).
(d) ?- bea(a,b) = bea(b,a).
(e) ?- ann(a,a) = bea(a,a).
(f) ?- ann(Ann,0) = ann(1,Ann).
(g) ?- bea(ann(0,1)) = bea(ann(Ann,Ann)).
(h) ?- ann(Ann) = 'ann(Ann)'.
(i) ?- bea(Bea) = [bea,Bea].
(j) ?- ['ANN'] = 'ANN'.
(k) ?- [bee,[]] = [bee].
(l) ?- [bee,[bee]] = [bee,[bea]].
(m) ?- [bea,[BEA]] = [bea,bea].
(n) ?- [[bea]] = [[[bea]]].
(o) ?- [ann,bea,[cee]] = [ann,bee,[cee]].
(p) ?- [ann,bee|[cee]] = [ann,bee,[cee]].
(q) ?- [[ann],bea|[BEA|ANN]] = [ANN,CEE].
(r) ?- [ann,cee,bee] = sort([cee,bee,ann]).
(s) ?- member(X,[ape,bee,cee]) = bee.
(t) ?- member(X,[ann,bee,cee]) = member(d,[ann,bee,cee,dee]).
```

Question 2. **Cutting it fine** (20 points)

Below you see a Prolog database with a cut-free definition of `pos/2`, whose both arguments are (possibly empty) lists of numbers. **Explain what this predicate does by discussing each clause, and give an example query that illustrates what it does**.

```
pos([X|L],[X|P]):- X > 0, pos(L,P).
pos([X|L],P):-  \+ X > 0, pos(L,P).
pos(L,P):- L=[], P=[].
```

Now we decide to add **one** cut (i.e., the built-in predicate `!/0`) to the definition. We consider four different positions for this cut, as shown below. For each of these four possible positions, **explain why this is a good or bad position** for the cut.

cut position 1

```
pos([X|L],[X|P]):- !, X > 0, pos(L,P).
pos([X|L],P):-      \+ X > 0, pos(L,P).
pos(L,P):-  L=[], P=[].
```

cut position 2

```
pos([X|L],[X|P]):- X > 0, !, pos(L,P).
pos([X|L],P):-  \+ X > 0, pos(L,P).
pos(L,P):- L=[], P=[].
```

cut position 3

```
pos([X|L],[X|P]):- X > 0, pos(L,P), !.
pos([X|L],P):-  \+ X > 0, pos(L,P).
pos(L,P):-  L=[], P=[].
```

cut position 4

```
pos([X|L],[X|P]):-    X > 0, pos(L,P).
pos([X|L],P):- !, \+ X > 0, pos(L,P).
pos(L,P):- L=[], P=[].
```

Question 3. **Count noses** (16 points)

Write a predicate `count/2` that counts elements in a list, but in a special way. If an element is directly repeated in the list it increases the count of that element. If the next element in the list is different, it starts counting from one again for the new element. Some examples should make this clear:

```
?- count([a,a,a,a,b,b,b,c,c,a],C).
C = [a:4,b:3,c:2,a:1]
yes

?- count([b,b,c,c],C).
C = [b:2,c:2]
yes

?- count([b,c,c,b,b,b],C).
C = [b:1,c:2,b:3]
yes

?- count([a,a,a,a,a],C).
C = [a:5]
yes
```

You can assume that in queries the first argument is a list with atoms, and the second argument is a variable.

Question 4. **The Young Ones** (22 points)

Given is the following database of people and their ages:

```
age(ann, 20).          age(joe, 44).
age(bob, 40).          age(min, 27).
age(cai, 30).          age(ned, 27).
age(deb, 42).          age(pat, 36).
age(edo, 24).          age(tod, 56).
```

This database asserts that the person named "ann" is 20 years old, "bob" is 40, and so on. Now answer the following questions:

(a) Write a predicate `older/2` that is true if and only if both arguments unify with a person in the database such that the first person is older than the second.

(b) Write a predicate `hundred/1` that is true if and only if its argument unifies with a list of (at least two) persons that have a cumulative age of 100.

(c) Write a predicate `middle/1` that is true if and only if its argument is a person that is not the youngest and not the oldest.

(d) Write a predicate `youngish/1` that is true if and only if its argument is a person that is not the youngest and not the oldest, but the number of older people is larger than the number of younger people.

All definitions need to be general, so if the database changes and other people and their ages are added, they should still work correctly. You may make use of the built-in predicates `findall/3`, `setof/3`, or `bagof/3`.

Question 5. **Taking the train** (10 points)

Consider the following Prolog database:

```
train(groningen,delfzijl).
train(groningen,leeuwarden).
train(groningen,assen).

direct(X,Y) :- train(X,Y).
direct(X,Y) :- train(Y,X).

route(X,Y,[X,Y]) :- direct(X,Y).
route(X,Y,[X|R]) :- route(Z,Y,R), direct(X,Z).
```

(a) How many facts are in this database?

(b) How many rules are in this database?

(c) How many clauses are in this databases?

(d) Which predicates are defined in this database?

(e) How many dynamic predicates are defined in this database?

(f) How many recursive predicates are defined in this database?

(g) How many tail-recursive predicates are defined in this database?

(h) How can you re-write the definition of `direct/2` in one rule?

Question 6. **The negative ones** (12 points)

Consider the following definition for the predicate `negative/2`.

```
negative(L,N):- L=[], N=[].
negative([X|L],[X|N]):- X < 0, !, negative(L,N).
negative([X|L],N):-     X > 0,    negative(L,N).
```

Draw the complete search tree for the query `?- negative([4,-3,-1,2],Neg).` Beware of the cut, it might remove branches from the search tree. When it does, indicate this clearly.

This exam has 6 questions. Total number of points: 100.
Dit tentamen bevat 6 opgaven. Puntentotaal: 100.