Tentamen Logisch Programmeren (LIX003B05)
27 January 2021, 3–6 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. **Fail at the first hurdle** (20 points)

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

(a) `?- ann = bea.`
(b) `?- bea = bea(Cee).`
(c) `?- bea(0) = bea(0,0).`
(d) `?- bea(0,1) = bea(0,0).`
(e) `?- ann(0,0) = bea(0,0).`
(f) `?- ann(Ann,0) = ann(1,Ann).`
(g) `?- bea(ann(0)) = bea(ann(Ann,Bea)).`
(h) `?- ann(Ann) = 'ann(Ann)'.`
(i) `?- bea(Bea) = [bea(Bea)].`
(j) `?- [ANN] = 'ANN'.`
(k) `?- [bea,[]] = [bea].`
(l) `?- [bea,[bea]] = [bea,bea].`
(m) `?- [bea,[BEA]] = [bea,bea].`
(n) `?- [[bea]] = [bea].`
(o) `?- [ann,bea] = [ann,bea|[_]].`
(p) `?- [ann,bea|[cee]] = [ann,bea,[cee]].`
(q) `?- [[ann],bea|[BEA|ANN]] = [ANN,CEE].`
(r) `?- [ann,cee,bea] = sort([cee,bea,ann]).`
(s) `?- member(X,[ape,bea,cee]) = ann.`
(t) `?- member(X,[ann,bea,cee]) = member(d,[ann,bea,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(L,P):- L=[], P=[].
pos([X|L],[X|P]):- X > 0, pos(L,P).
pos([X|L],P):-  \+ X > 0, pos(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 show below. For each of these four possible positions, **explain why this is a good or bad position** for the cut.

cut position 1

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

cut position 2

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

cut position 3

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

cut position 4

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

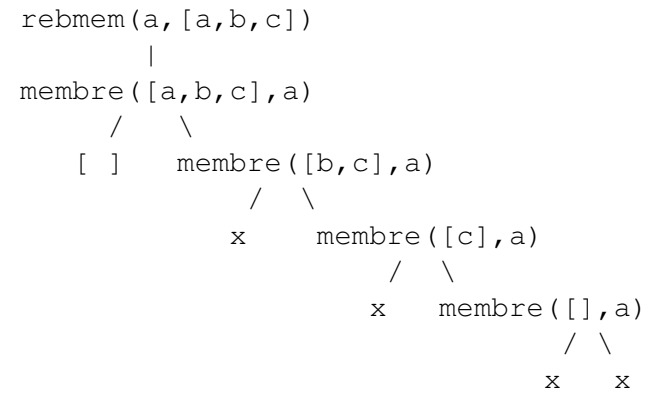
Question 3. **Diversity matters** (10 points)

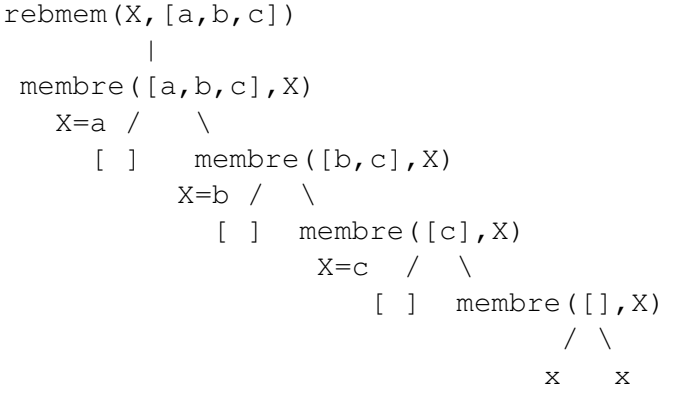
Write a predicate `diversity/1` that is true if and only if (a) its argument (a Prolog list) is a list with at least three members, and (b) contains elements that are all different from each other. You may use the standard member/2 Prolog predicate in your definition, or add other (standard) auxiliary predicates. Some example queries that show the intended behaviour of this predicate are:

```
?- diversity([a,b,c]).
yes

?- diversity([a,b,a,b]).
no

?- diversity([a,b,b,a,b]).
no

?- diversity([a,c,b,d]).
yes

?- diversity([a,d]).
no
```

Question 4. **Tree Analysis** (14 points)

Consider the following Prolog search trees drawn from a certain Prolog database for the predicates `membre/2` and `rebmem/2` (the `x` denotes a "dead" branch, a branch without a solution, and `[ ]` shows a succesful branch)). Based on these search trees, give the definitions for both predicates.





Question 5. **The Young Ones** (16 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, 33).
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 `younger/2` that is true if and only if both arguments unify with a person in the database such that the first person is younger than the second.

(b) Write a predicate `same_age/2` that is true if and only if its argument unifies with a list of (at least two) persons that have the same age, and the second argument unifies with the age of these persons. Hint: use one of the three built-in predicates `findall/3`, `bagof/3` or `setof/3` in your definition.

(c) Write a predicate `oldest/1` that is true if and only if the first argument is a person that is older than all other people in the database.

All definitions need to be general, so if the database changes and other people and their ages are added, they should still work correctly.

Question 6. **Taking the train** (8 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 clauses are in this databases?

(b) How many rules are in this database?

(c) How many facts are in this database?

(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?

Question 7. **A train of thought** (12 points)

Given the Prolog database of the previous question (*Taking the train*), draw the complete search tree for the query

```
?- route(delfzijl,assen,R), !.
```

Be aware of the cut; it will remove some of the branches!

This exam has 7 questions. Total number of points: 100.
Dit tentamen bevat 7 opgaven. Puntentotaal: 100.

Docent: Johan Bos

Tweede lezer: Gertjan van Noord