Logisch Programmeren en Zoektechnieken: Tentamen A **ANSWERS**
22 September 2014

Instructions: This exam paper consists of four questions. You may answer them in Dutch or in English. Only write in the answer booklets provided and mark each booklet with your name and student number. Give short and precise answers. Write clearly. The maximum time allowed is 120 minutes.

Question 1

(a) For each of the following two built-in Prolog predicates, give a short explanation of their functionality: 5 marks

- `member/2`
- `append/3`

Support each of your explanations with an example.

Answer:

See Lecture Notes.

(b) A *palindrome* is a word that does not change when read from back to front, such as *madam* or *rotator*. Implement a Prolog predicate called `palindrome/1` that checks whether a given Prolog atom (representing a word) is a palindrome. Examples: 10 marks

```
?- palindrome(madam).
Yes

?- palindrome(sir).
No
```

Hint: Make use of appropriate built-in predicates.

Answer:

```
palindrome(Word) :-
  atom_chars(Word, Letters),
  reverse(Letters, Letters).
```

(c) Implement a Prolog predicate called `middle/2` that, when given a list with an odd number of elements in the first argument position, will return the middlemost element from that list in the second argument position. Examples: 10 marks

```
?- middle([lion, puma, tiger, eagle, crocodile], X).
X = tiger
Yes

?- middle([1, 2, 3, 4, 5, 6, 7], Y).
Y = 4
Yes
```

Answer:

```
middle([Mid], Mid).

middle(List, Mid) :-
  append([_|Rest], [_], List),
  middle(Rest, Mid).
```

Question 2

(a) Implement a Prolog predicate called `count_atoms/2` that, when given a list of terms in the first argument position, will return the number of atoms in that list in the second argument position. Examples: [10 marks]

```
?- count_atoms([a, 2, b, f(7), b, b, c], Number).
Number = 5
Yes

?- count_atoms([X, Y, Z], Number).
Number = 0
Yes
```

Answer:

```
count_atoms([], 0).

count_atoms([Atom|Tail], N) :-
  atom(Atom), !,
  count_atoms(Tail, N1),
  N is N1 + 1.

count_atoms([_|Tail], N) :-
  count_atoms(Tail, N).
```

(b) Implement a Prolog predicate called `divisors/2` that, when given a positive integer in the first argument position, will return the list of all the divisors of that integer in the second argument position. Example: [10 marks]

```
?- divisors(30, List).
List = [1, 2, 3, 5, 6, 10, 15, 30]
Yes
```

Answer:

```
divisors(N, List) :-
  divisors(N, 1, List).

divisors(N, K, []) :-
  K > N, !.

divisors(N, K, [K|List]) :-
  N mod K =:= 0, !,
  K1 is K + 1,
  divisors(N, K1, List).

divisors(N, K, List) :-
  K1 is K + 1,
  divisors(N, K1, List).
```

(c) For each of the following five queries, say whether it would succeed, fail, or result in an error message. If it succeeds, also provide all variable instantiations. You only need to account for the *first* answer given by Prolog. [5 marks]

(*i*) `?- .(a, .(b, [])) = [_|X].`

Answer:

```
?- .(a, .(b, [])) = [_|X].
X = [b]
Yes
```

(*ii*) `?- [X] = [X|X].`

Answer:

```
?- [X] = [X|X].
X = []
Yes
```

(*iii*) `?- append(_, [X|Y], [a,b,c,d,e,f,g]), length(Y, 5).`

Answer:

```
?- append(_, [X|Y], [a,b,c,d,e,f,g]), length(Y, 5).
X = b,
Y = [c, d, e, f, g]
Yes
```

(*iv*) `?- X = 12 // 3.`

Answer:

```
?- X = 12 // 3.
X = 12 // 3
Yes
```

(*v*) `?- X == 12 // 4.`

Answer:

```
?- X == 12 // 4.
No
```

Question 3

(a) Explain how you can define your own operators in Prolog. Your explanation should cover the features defining an operator, the built-in predicate to be employed, and the practicalities of integrating the operator definition into a program. [15 marks]

Answer:
See Lecture Notes. A complete answer should cover operator names, precedence values, associativity patterns, `op/3`, and query execution at compilation time.

(b) Provide operator definitions that ensure that the three queries below all succeed: 10 marks

```
?- bli bli blibli bla bli blibli = bli X bla X.
?- a bla b bla c \= (a bla b) bla c.
?- blabla bla blabla blu = blu(_).
```

Answer:

```
:- op(100, fy, bli),
   op(200, xfy, bla),
   op(300, xf, blu).
```

Question 4

(a) Explain how you can implement an operator for *negation as failure* in terms of a *cut* (`!`). Provide both a suitable operator definition and a definition of the corresponding predicate. 5 marks

Answer:

See Lecture Notes.

(b) Consider the following Prolog program: 10 marks

```
p([], []).
p(List, [X|Perm]) :- select(X, List, Rest), p(Rest, Perm).

o([]).
o([_]).
o([A,B|Rest]) :- A =< B, o([B|Rest]).

s(List, SList) :- p(List, SList), o(SList).
```

The predicate `s/2` can be used to sort a given list of numbers:

```
?- s([7, 3, 2, 8, 1, 4, 6, 5], Result).
Result = [1, 2, 3, 4, 5, 6, 7, 8]
Yes
```

Explain how this program solves the problem of devising a general method for sorting a given list of numbers. What is the overall idea underlying the solution given? What is the role of the predicates `p/2` and `o/1`? When you explain the program, make sure you include at least a brief comment on each of its clauses.

Answer:
The predicate `p/2` succeeds if the second argument can be instantiated with a permutation of the list given in the first argument position (see Lecture Notes). The predicate `o/1` succeeds if the list of numbers provided is in ascending order (see Bratko). The overall idea is to use backtracking to generate all permutations of the input list, until one of them is found to be ordered. (This should be followed by a brief explanation of how the three predicates are implemented.) Note that this is a very slow method for sorting lists of numbers; it is only usable for lists of length up to about 10.

(c) Write a Prolog predicate `find/2` to check whether a given number can be found anywhere within a given compound term. Examples: 10 marks

```
?- find(5, f(5)).
Yes

?- find(1, f(2,3)).
No

?- find(3, g(5,f(f(3,5)),7)).
Yes
```

Answer:

```
find(X, X).

find(X, Term) :-
  Term =.. [_|Args],
  member(Arg, Args),
  find(X, Arg).
```