NAME:

Examination of
Programming languages and Programming Methodologies
10 January 2020, 8h00 – 12h00

General Guidelines

The examination is **closed book** and takes (at most) 4 hours.
Make sure that your handwriting is **readable**.
Write your name on every sheet you hand in.
Write on **one** side of the sheet only.

Questions

1. Do the following Prolog queries succeed or fail? In case of success, give the bindings of the variables in the query. In case of failure, explain why.

   1) `?- [X,Y|T] = [[b,Z],[],[Z]] .`
   2) `?- member(3 + Y, 7) = member(A, B), Y = 4, A is B`
   3) `?- L = [a,[c],4|T], L-T = A-[].`

   What is the domain of the decision variable Z in CLP(FD) after the following query:

   ```
   ?- Y in 1..7, X in -3..100, Y #> X, Z #\= 0,  Z #= Y - X.
   ```

2. Consider the following Prolog predicate.

   ```
   element(X, [X | _T]).
   element(X, [Y | _T]) :-  element(X, Y).
   element(X, [_Y | T]) :-  element(X, T).
   ```

   Does the query `?-element(R,[1,[3,S],4]).` succeed or fail? Sketch its execution by Prolog (e.g. by giving an execution tree). In case of success, indicate how many times it succeeds and give the bindings of the variables (if any) in the query. In case of failure, explain why.

3. Consider the following puzzle. Take the twelve numbers on the face of a clock, namely 1, 2, 3, ..., 11, 12. We want to rearrange the numbers (keeping them in a circle) in such a way that no triplet of adjacent numbers has a sum higher than 21.

   Write the predicate `clock_round(N, Sum, Xs)` that can solve the above puzzle and related ones. For the above puzzle the call is `?- clock_round(12, 21, Xs)`. It finds a lot of solutions for Xs (e.g. [12,2,6,11,4,5,9,7,3,10,8,1], [12,2,7,9,5,4,11,6,3,10,8,1], ..., and [12,7,2,11,6,4,10,5,3,9,8,1]).

   If you take the numbers 1 upto 5, the call `?- clock_round(5, 10, Xs)` finds [5,2,3,4,1] and [5,3,2,4,1] as solutions for Xs. Note that all rotations of [5,2,3,4,1] are also valid solutions e.g [2,3,4,1,5], [3,4,1,5,2], ... , as is [5,1,4,3,2] (when you read the numbers anti-clockwise) and its rotations. Note also that [5,2,3,1,4] is not a solution, because 4+5+2 is larger than 10.

   First write a version of `clock_round/3` that solves the puzzles. In a next step, try to avoid solutions that are just variants (e.g. the rotations and the anti-clockwise readings).

   Indicate clearly whether you are using CLP(R), CLP(FD) or just normal Prolog.

4. You are given a set of items. Each item has a name, a weight and a value. For example, you have the following 4 items: an ax with weight 50 and value 40, a book of Norvig with weight 50 and value 50, a box of cookies with weight 10 and value 5, and a laptop with weight 99 and value 60.

   1) Represent the given information in two different ways: by Prolog facts and by Prolog terms. Use your 2 representations to represent the above set of 4 items.

   2) You have to compute all subsets of items (containing at least one element) such that their total weight is strictly smaller than a given weight W and their total value is strictly higher than a given V.

      In the case of our example there are 3 possible subsets that have a weight strictly smaller than 90 and value strictly higher than 40. namely 1) ax and box of cookies, 2) book of Norvig and box of cookies and 3) book of Norvig.

Choose one of the 2 representations and define the predicate **subset** to compute such subsets. Note that in general there can be a finite number n of such items.

3) Write a predicate `highest` to find the list of all items with the highest value. You can again choose one of the 2 representations.

5. We have to solve the **exact covering** problem. In this problem there is a given universe of items and a given set of options. Each option is a set items from the universe.

   Consider an example with as universe the 7 items a,b,c,d,e,f, and g and 6 options, namely option (1) consisting of c and e, option (2) consisting of a, d, and g, option (3) consisting of b, c, and f, option (4) consisting of a, d, and f, option (5) consisting of b and g, option (6) consisting of d, e, and g.

   The goal of an exact covering problem is to find disjoint options that cover all items in the universe.

   In the above example, option (1), (4) and (5) are disjoint and their union give us each item exactly once.

   Every tentative choice we make leaves us with a residual exact cover problem that is smaller. Suppose we try to cover item a by choosing the option (2) consisting of a, d, and g: The residual problem has only two options, namely (1) with c and e and (3) with b, c, and f because the other four involve the already-covered items. Now it is easy to see that options (1) and (3) have no solution (that covers b,c,e, and f): option (1) and (3) are not disjoint as they both contain item c. Therefore, we can remove option (2). That leaves us with only one option for item a, namely (4) with a, d, and f and its residual consisting of options (1) and (5) to cover b,c,e, and g.

   1) How do you represent in Prolog the universe of items to be covered? What is your representation for the universe in the example?
   2) How do you represent in Prolog the set of options? What is your representation for the options in the example?
   3) An item is covered by an option if that option contains the item. Given a set of options Os and an item I to be covered write a predicate `iscovered(I, Os, O)`. This predicate fails if I can not be covered by the options in Os and succeeds with O being a covering option.

4) Given a set of items Is including I and given a set of options Os including a covering option O for I, write a predicate `residual` to determine the residual items and options while using O to cover I.

5) Write a predicate `findexactcovering` that determines for a given universe of items and a given set of options an exact covering (as a subset of options). It fails if no exact covering can be found.

Good luck.

Gerda Janssens