Logic and Constraint Programming
Exam – November 6, 2017 – 2h

At the end of the exam, you must print your program and give it to the exam supervisor.
(In case of printing problem, send it, in one file, to `jan-georg.smaus@irit.fr` and `jerome.mengin@irit.fr`.)

Your programs must be well commented: for each predicate that you define, describe the kind of inputs/outputs, and when the predicate is true.

You can use gprolog to test your programs.
You can use the prolog manual at `www.gprolog.org/manual/gprolog.html`. No other document is authorized.

You MUST NOT*:*

- *use an internet browser – except to consult the prolog manual - nor use other communication tools (mail, chat, forums,...)*
- *consult other person's programs.*

Exercise 1 (Nests,7 points)**.** Un **nest** ("nid") est une certaine structure de données : un arbre dans lequel le nombre de fils d'un nœud est indéterminé. On peut l'utiliser pour stocker des données arbitraires, par exemple des entiers. Un nest est réalisé en utilisant une liste pour les fils d'un nœud. Formellement :

- Pour tous entier k, e(k) est un nest;
- Si $a_1,\dots,a_m$ sont des nests, alors n([$a_1,\dots a_m$]) est un nest.

Par exemple, `e(4)`, `n([e(4),e(7),e(2)])`, `n([e(3),e(0),e(1),e(8)])`, `n([n([e(4),e(7),e(2)]),n([e(3),e(0),e(1),e(8)])])` sont des nests.

Question 1.1. Écrivez un prédicat `is_nest/1` qui sera vrai si l'argument est un nest (on pourra aussi l'appeler avec une variable non-instantié, dans ce cas le prédicat devrait énumérer des nests). Par exemple, `is_nest(n([e(3),e(0),e(1),e(8)]))` devrait donner le réponse "true" et `is_nest(n([n(3),e(0),e(1),e(8)]))` ou `is_nest([1,2,3])` devraient donner "false".

Question 1.2. Écrivez un prédicat `mirror/2` pour renverser un nest. Par exemple, l'appel `mirror(n([n([e(4),e(7),e(2)]),n([e(3),e(0),e(1),e(8)])]),N)` doit instantier `N` comme `n([n([e(8),e(1),e(0),e(3)]),n([e(2),e(7),e(4)])])`.

Question 1.3. Écrivez un prédicat `flatten/2` pour "aplatir" un nest, c'est-à-dire, pour convertir le nest dans une liste qui contient les éléments dans les feuilles d'un nest dans un passage gauche-à-droit. Par exemple, l'appel `flatten(n([n([e(4),e(7),e(2)]),n([e(3),e(0),e(1),e(8)])]),L)` doit instantier L comme `[4,7,2,3,0,1,8]`.

For the next two exercises, you can use all built-in predicates of gprolog, including those for the finite-domain constraint solver. You must define all other predicates that you use.

Exercise 2 (6 points)**.** Assume a weighted directed graph is described by means of a predicate edge/3, such that edge(X, Y, C) is true is there is an edge from vertex X to vertex Y of cost C. For instance, to the right is a graph and its description using edge/3:

edge($a, c, 1$).
edge($a, d, 3$).
edge($b, d, 2$).
edge($c, e, 5$).
edge($e, c, 2$).
edge($e, f, 2$).
edge($d, f, 10$).

$$\begin{tikzcd}
a \arrow[r, "1"] \arrow[dr, "3"] & c \arrow[r, bend left, "5"] & e \arrow[l, bend left, "2"] \arrow[d, "2"] \\
b \arrow[r, "2"'] & d \arrow[r, "10"'] & f
\end{tikzcd}$$

Question 2.1. Define a predicate cheaperPath/3, such that cheaperPath(X, Y, N) is true if there is a path from X to Y of total cost less than N. The predicate is supposed to be called with X, Y and N all instantiated, for example to the query cheaperPath($a, f, 7$) the anwer should be no.

Question 2.2. Define a predicate reachable/2 that computes the list of nodes that can be reached from a given node. For instance, to the query reachable(a, L) Prolog should answer L=[c, e, d, f] (in any order). *(Remember findall.)*

Exercise 3 (7 points)**.** The organiser of an event must choose rooms for 6 seminars that will be held on the same day. He has 4 rooms, numbered 1 to 4. Each seminar is composed of presentation that have already been scheduled. The table on the right shows the times of the presentations for the 6 seminars, noted "a" to "f". No seminar can change room, and, of course, there must not be 2 presentations from 2 different seminars at the same time in the same room.

sem.	9h	10h	11h	12h	13h	14h	15h
a		X	X	X			
b		X	X				
c	X	X	X				
d				X	X	X	
e				X	X		
f					X	X	X

Question 3.1. Write a simple program, using the constraint solver of gprolog, that computes an allocation of rooms for the seminars. In particular, you must define a predicate named solution, such that solution(A, B, C, D, E, F) is true if A, B, $\ldots F$ are the numbers of the rooms chosen for each seminar.

Question 3.2. Write now a program that finds solutions that minimize the number of rooms that are necessary, and also returns the number of rooms used.

The director of a big hotel has a similar problem: allocate rooms for the customers, depending on their arrival and departure dates. But the dates are not given in a small table like the one above, but in a database that contains hundreds of prolog facts:

client(N, A, D) indicates that the client named N arrives on day A and leaves on day D.

Each date is a number between 1 et 31 : we consider that the hotel only exists for one month (to make things a little simpler)!

The hotel has 70 rooms, numbered from 1 to 70. Two clients must not share the same room. But a client that arrives on a given day can use a room that has been used by another client who leaves on that day.

Question 3.3. Write a prolog program, again using the constraint solver, that computes an allocation of the rooms to the clients. Try to minimize the number of rooms that are necessary, since, if a room is not used during a month, it saves some cleaning costs.