Write predicates to evaluate propositional programs, in particular: the least model for definite logic programs.

This assignment will give you training on writing "interpreters" in Prolog. The assignment is structured in a sequence of steps: ranging starting from simpler basic ones that are used to construct the subsequent, more complex ones.

Propositional programs manipulated by your program will be given by a list where each element is of the form rule(p, b) where p is a proposition and b is a (possibly empty) list of literals. Each literal is a proposition (i.e. a positive literal).

Example propositional program and its representation:

(a) p <- r.
q <- q.
r.

[rule(p, [r]),

rule(q, [q]),

rule(r, [])]

Write a predicate propositions(P, L) that finds the list L of all propositions in the given propositional program P. For example, when P is the example program (a), propositions(P, L) should succeed with L = [p,q,r].

For the above question, the list (the answer) should not have duplicates; the order of elements in the list does not matter.

Write a predicate tp(P, M1, M2) that, given a list of predicates M1 finds a list M2 such that M2 is the immediate consequence of the propositional program P with respect to M1: i.e. M2 = TP(M1).

For this predicate, assume that the given propositional program is definite.

For instance, when P is the example program (a), tp(P, [r], M) will succeed with M = [p,r].

Use library predicate sort/2 to ensure that the list M2 will always be given in the same order.

Write a predicate leastmodel(P, M) that returns a list M of propositions in the least model of the given propositional definite program P. For instance,

• when P is the example program (a), leastmodel(P, M) will succeed with M = [p, r].