In this assignment, you will write a simplified version of a Prolog interpreter in OCaml.

You will first define an ML data type to represent the structure of a legitimate Prolog program.

* A program is a set (list) of clauses.
* A clause can either be a fact or a rule. A fact has a head but no body.  A rule has a head and a body.
* The head is a single atomic formula.  A body is a sequence of atomic formulas.
* An atomic formula is a k-ary predicate symbol followed by k terms.
* A term is either a variable, a constant, or a k-ary function symbol with k subterms.
* A goal is a set (list) of atomic formulas.

You need to take your implementation of unification to use as the parameter-passing mechanism. (Note: by pretending the predicate symbol is a function symbol, you can perform resolution of goals and program clauses).

You also need to develop a back-tracking strategy to explore the resolution search space.   You need to be able to replace a goal by subgoals, as found by applying a unifier to the body of a program clause whose head unified with the chosen subgoal.

Check that your interpreter can execute the family relationship programs you have specified earlier