PROLOG FACTS

```
old(bill).
age(anne,29).
father(john,mary).
king(henry,8,england,1509,1547).
date(thursday,15,february,1996)
book('Algorithms','Sedgwick,R','Addison-Wesley',1988)

age(jane,24).
age(jill,26).
age(julia,33).
age(mary,29).
age(alex,26).
age(arthur,26).
age(bill,33).
age(eric,17).
age(john,42).
```

PROLOG RULES

```
% A rule:
 get_age :-
         write('Enter person: '),
         read(P),
         age(P,A),
         write(P), write(' has age '), write(A).
```

Example

```
?- age(eric,X).
```
(succeeds, with `X = 17`).

```
?- age(alex,X).
```
(succeeds, with `X = 26`).

```
?- age(susan,X).
```
(fails).

```
?- age(P,17).
```
(succeeds, with `P = eric`).

```
?- age(P,19).
```
(fails).

```
?- age(P,26).
```
(succeeds, with `P = jill`).

```
?- X is 4 * (5 + 2).
?- length([3,7,1,4],L).
```

More Rules

```
father(james,mary).
mother(jane,brian).
parent(X,Y) :- father(X,Y); mother(X,Y).

father_age(Person,Age) :- father(Father,Person),
                          age(Father,Age).
?- father_age(mary,A).

get_father_age :- write('Enter person: '),
                  read(Person),
                  father_age(Person,Age),
                  write('The father of '),
            write(Person),
                  write(' has age '), write(Age).
 get_father_age :- write('Not known.').
```

Deliberate Backtracking (Failure-driven loop)

```
people_of_age(A) :-
          age(P,A),
          write(P), nl,
          fail.
  people_of_age(A) :-
          write('End of list').
```

Recursion

```
stars(0) :- nl.
stars(N) :- write('*'), nl, M is N-1, stars(M).

stars :- write('*'), nl, stars.

royal(victoria).
royal(X) :- parent(P,X), royal(P).

archer(dan).
archer(X) :- father(P,X), archer(P).
```

Lists and Patterns

```
[3,7,5,29,6,3,1,2]
[mary,john,bill,arthur]
[75]
[]

writelist([]).
writelist([H|T]) :-
            write(H), nl,
            writelist(T).
```

[H|T] means the list with head H and tail T.

Predicates as Functions

```
double(X,Y) :- Y is 2*X.

max(A,B,A) :- A >= B.
max(A,B,B) :- B > A.

sumlist([H|T],S) :- sumlist(T,N), S is H+N.
sumlist([X],X).

duplicate([],[]).
duplicate([H|T],[H,H|T1]) :- duplicate(T,T1).
```