```
declare(Var,Val) :-
        retract(binding(Var,_)),
        assert(binding(Var, Val)).
declare(Var,Val) :- assert(binding(Var,Val)).

bind(Variable,Value) :-
        retract(binding(Variable,_)),
        assert(binding(Variable, Value)).

valueOf(Variable,Value) :- binding(Variable,Value).

undeclare(Var) :- retract(binding(Var,_)).

inc(Variable) :-
        retract(binding(Variable,Value)),
        NewValue is Value + 1,
        assert(binding(Variable, NewValue)).

dec(Variable) :-
        retract(binfing(Variable,Value)),
        NewValue is Value - 1,
        assert(binding(Variable, NewValue)).

add(Variable,Number) :-
        retract(binding(Variable, Value)),
        NewValue is Value + Number,
        assert(binding(Variable, NewValue)).

displayBindings :-
        binding(Variable,Value),
        write(Variable),
        write(' => '),
        write(Value), nl, fail.
displayBindings.

prepend(Variable,Value) :- %assume a list!
        retract(binding(Variable,OldValue)),
        NewValue = [Value|OldValue],
        assert(binding(Variable,NewValue)).

add(Variable1, Variable2, Result) :-
        binding(Variable1, Value1),
        binding(Variable2, Value2),
        undeclare(Result),
        ResultValue is Value1 + Value2,
        declare(Result,ResultValue).

sub(Variable1, Variable2, Result) :-
        binding(Variable1, Value1),
        binding(Variable2, Value2),
        undeclare(Result),
        ResultValue is Value1 - Value2,
        declare(Result,ResultValue).

mul(Variable1, Variable2, Result) :-
        binding(Variable1, Value1),
        binding(Variable2, Value2),
        undeclare(Result),
        ResultValue is Value1 * Value2,
        declare(Result,ResultValue).

div(Variable1, Variable2, Result) :-
        binding(Variable1, Value1),
```

```
        binding(Variable2, Value2),
        undeclare(Result),
        ResultValue is Value1 / Value2,
        declare(Result,ResultValue).

pow(Variable1, Variable2, Result) :-
        binding(Variable1, Value1),
        binding(Variable2, Value2),
        undeclare(Result),
        ResultValue is Value1**Value2,
        declare(Result,ResultValue).
```