# Experiment no. 8

**Aim:** To find max of two number using Prolog.

**Requirements:** Compatible version of SWI-Prolog.

**Theory:**

There is a built-in predicate construction in Prolog which allows you to express exactly such conditions: the if-then-else construct. In Prolog, *if A then B else C* is written as ( A -> B ; C). To Prolog this means: try A. If you can prove it, go on to prove B and ignore C. If A fails, however, go on to prove C ignoring B. The max predicate using the if-then-else construct looks as follows:

```
max(X,Y,Z) :-
   (  X =< Y
   -> Z = Y
   ;  Z = X
    ).
```

## Prolog's Persistence

• When a subgoal fails, Prolog will backtrack to the most recent successful goal and try to find another solution.

• Once there are no more solutions for this subgoal it will backtrack again; retrying every subgoal before failing the parent goal.

• A call can match any clause head.

• A redo ignores old matches.

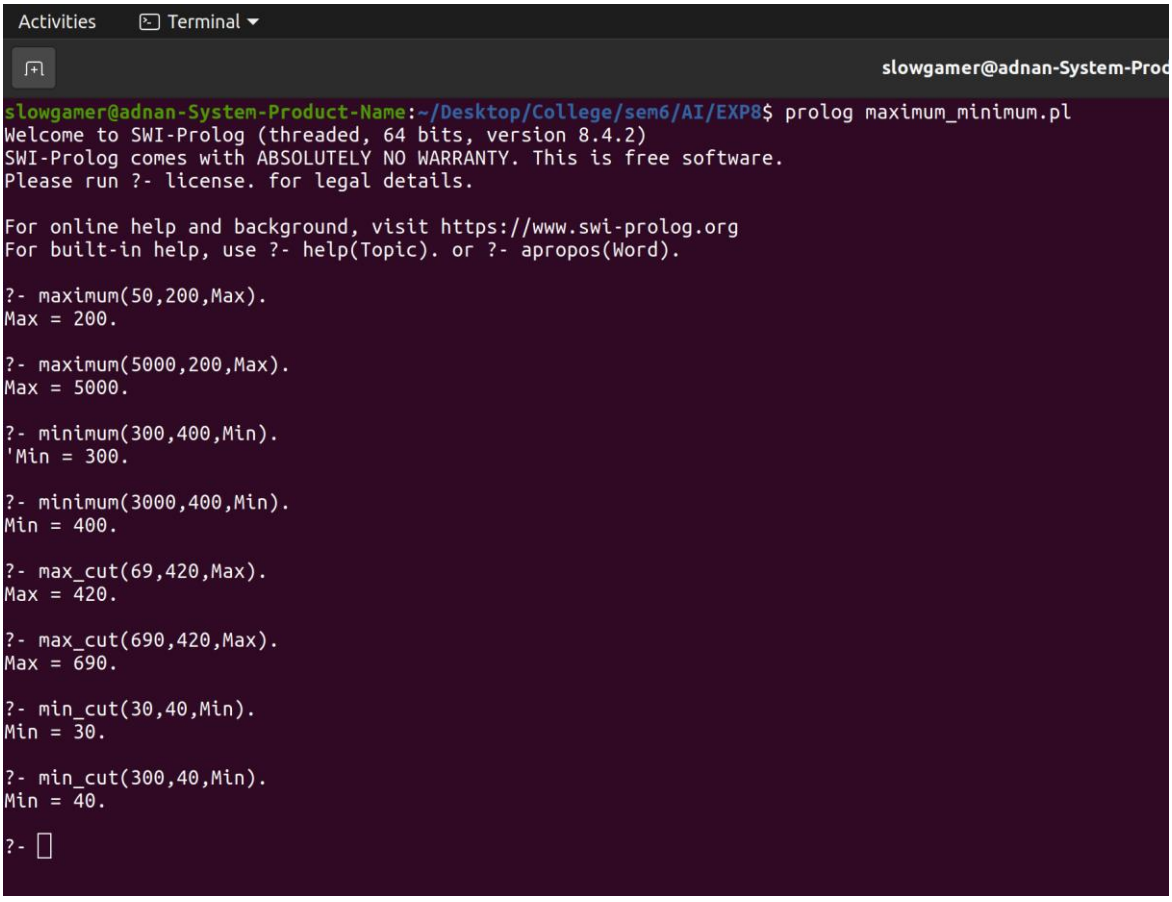
## Cut !

The cut, in Prolog, is a goal, written as !, which always succeeds, but cannot be backtracked past. It is used to prevent unwanted backtracking, for example, to prevent extra solutions being found by Prolog.

**Code:**

```
/*Without Cut*/

maximum(X,Y,Z):- (X>=Y-> Z=X; Z=Y).

minimum(X,Y,Z):- (X>=Y-> Z=Y; Z=X).

/*With Cut*/

max_cut(X,Y,Max):- X>=Y,!,Max=X; Max=Y.

min_cut(X,Y,Min):- X>=Y,!,Min=Y; Min=X.
```

**Output:**




**Conclusion:** We have successfully implemented Maximum number finding code in Prolog.