**CS 6364.002 ARTIFICIAL INTELLIGENCE**

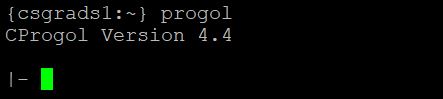
**Programming Assignment – 2**

**October 24, 2017**

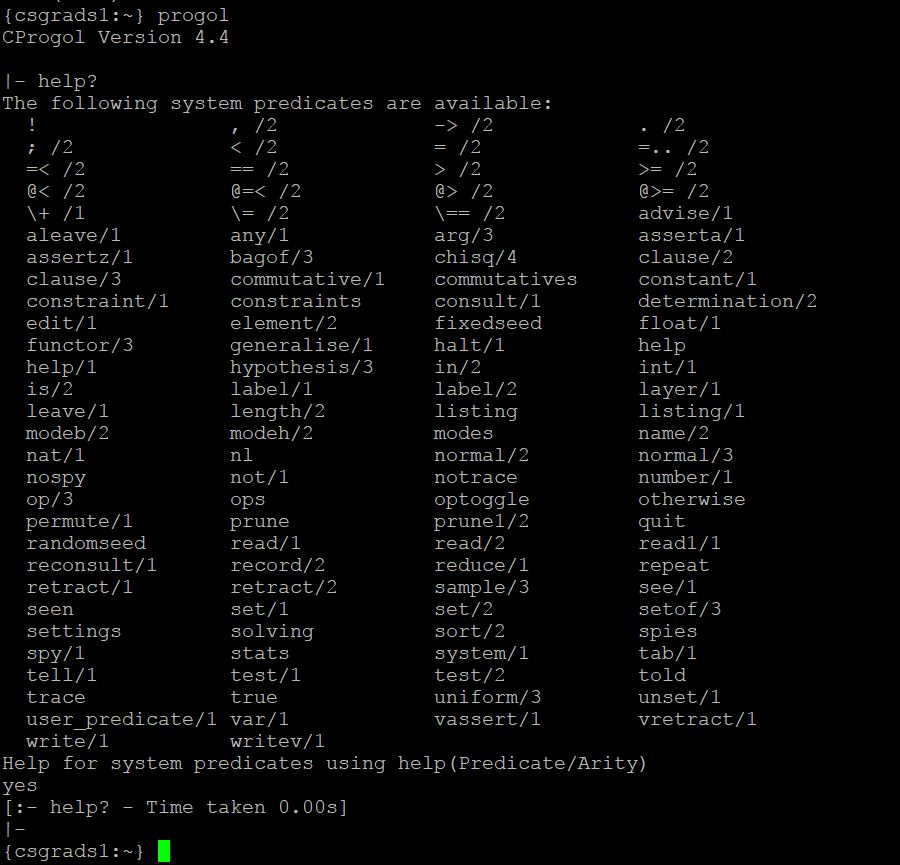
**Dileep Gudena (dxg161730)**

**Gokul Surendra (gxs161530)**

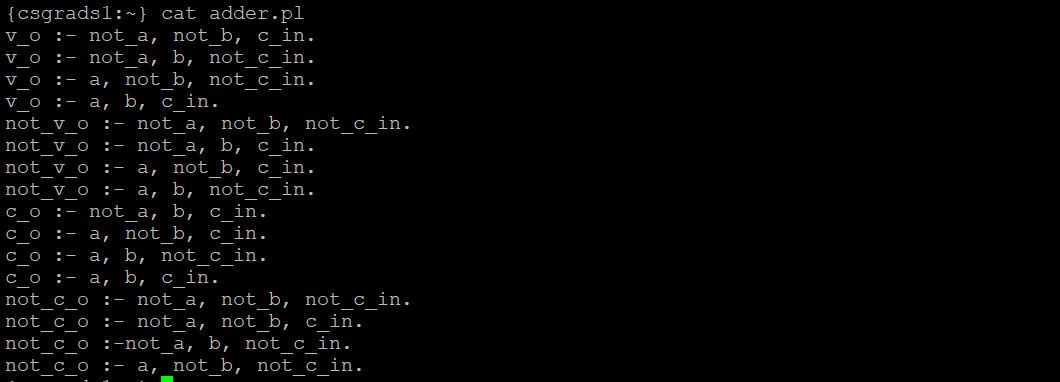
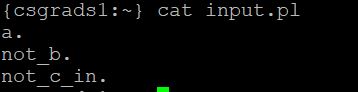
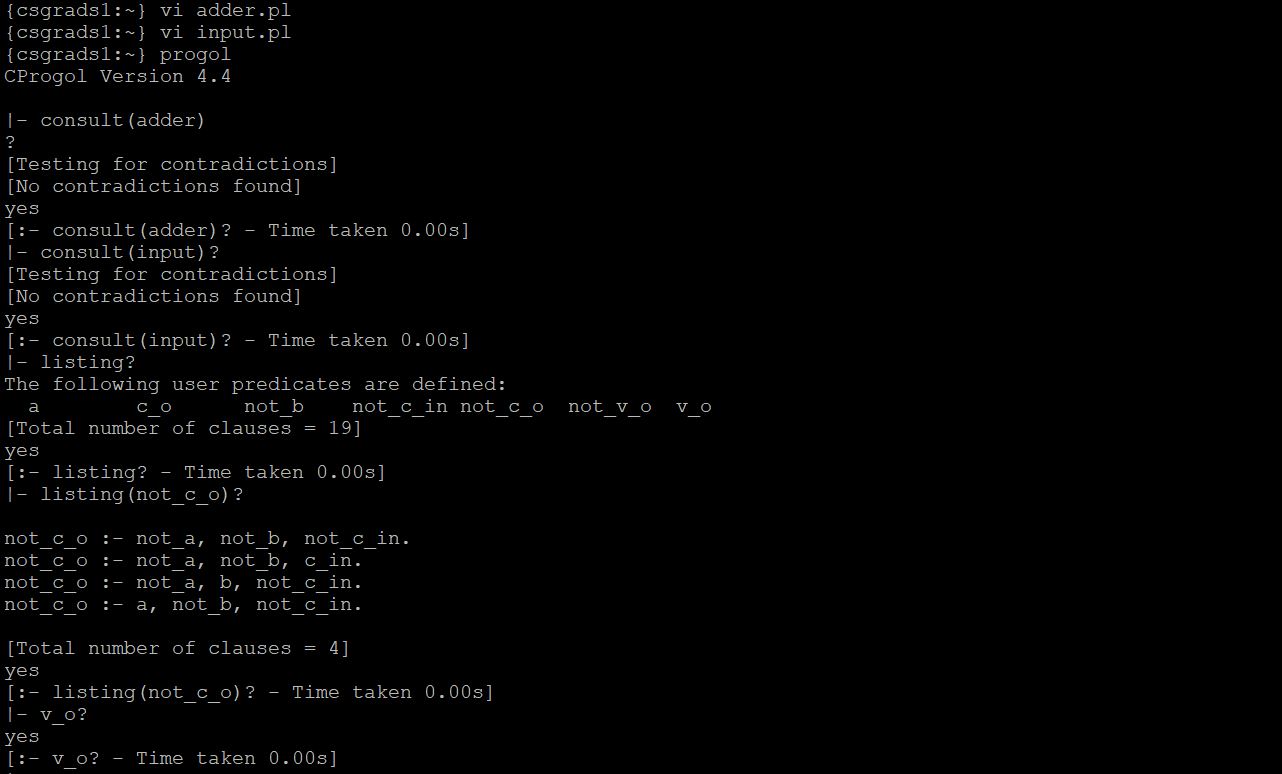
**Exercise 1:**



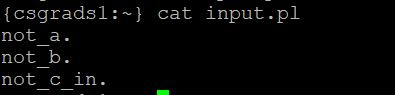
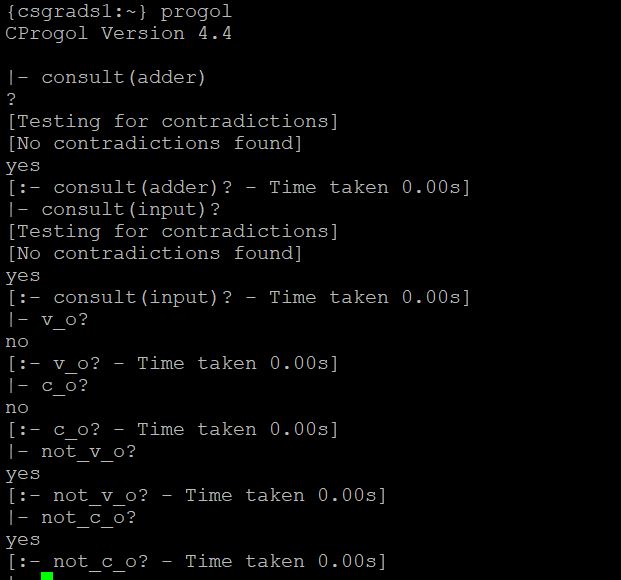
**Exercise 2:**

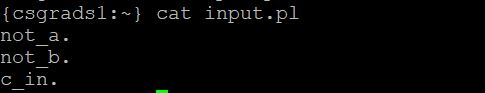
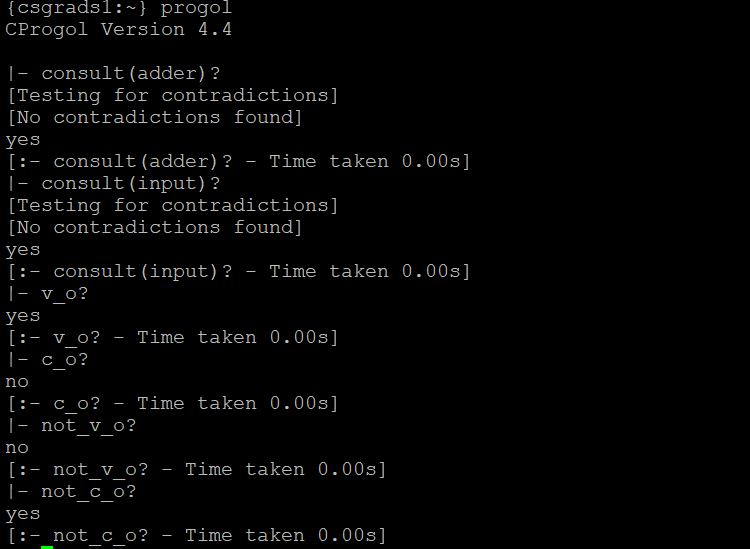


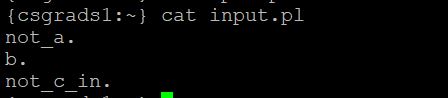
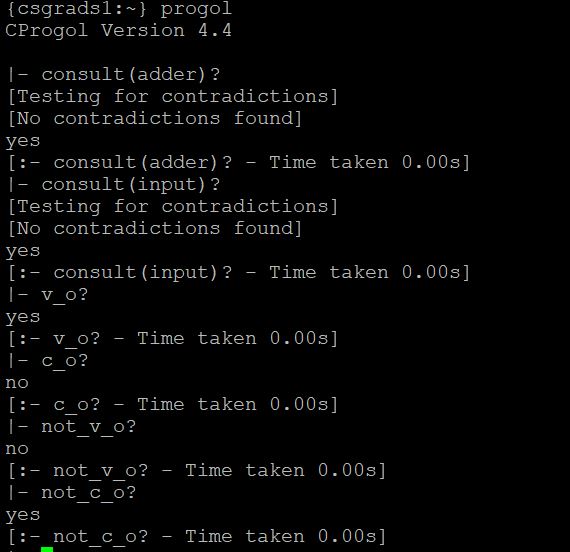
**Exercise 3:**

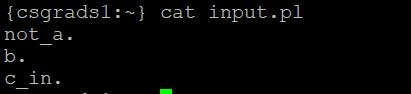
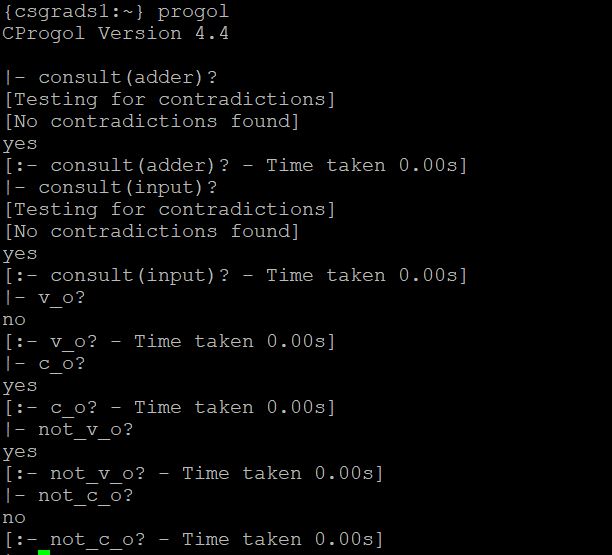
  

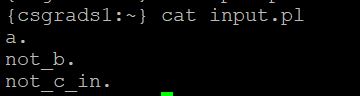
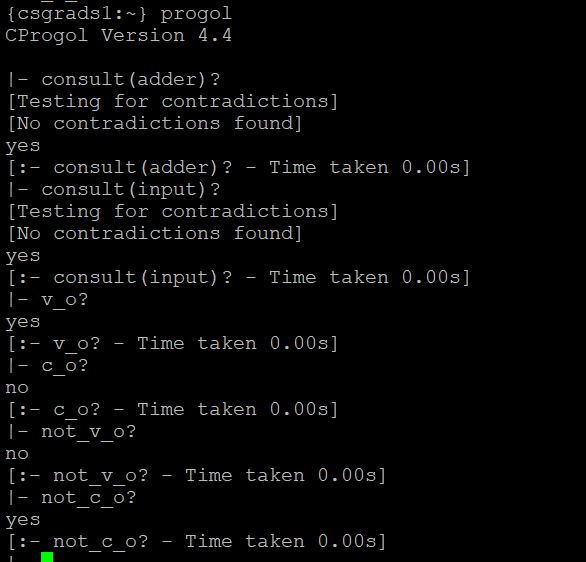
**Exercise 4:**

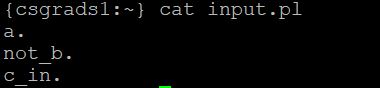
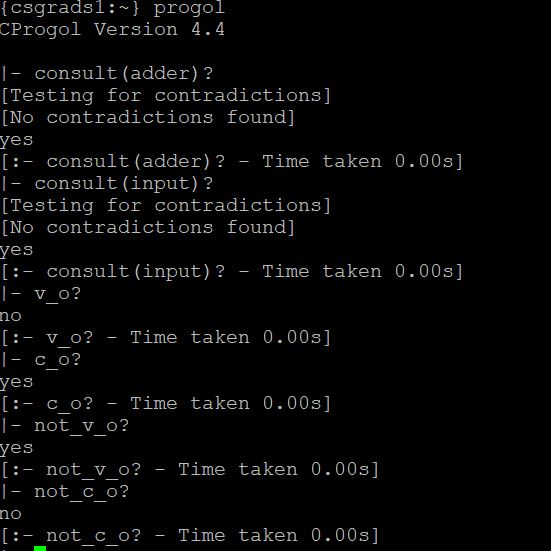
 

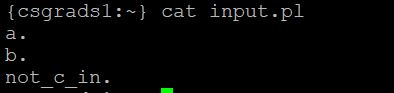
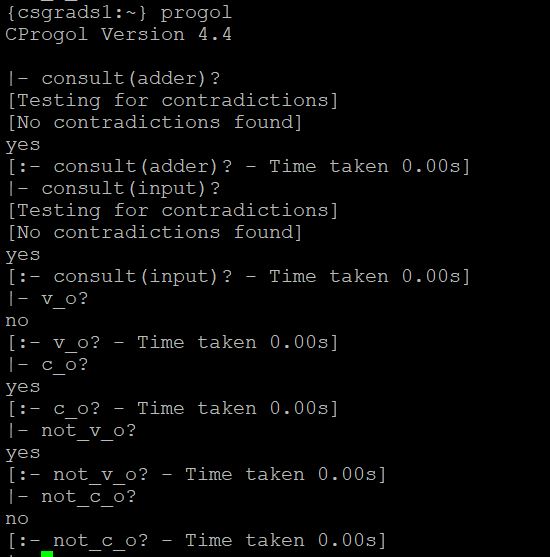
 

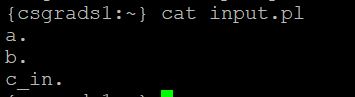
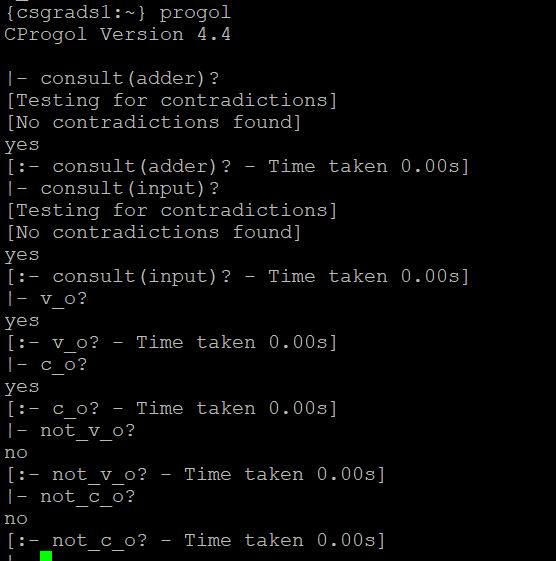
 

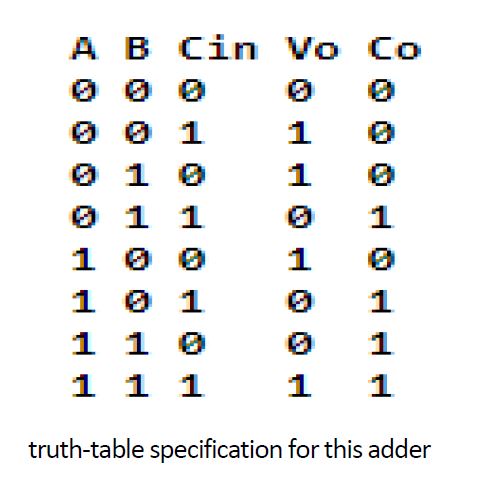
 

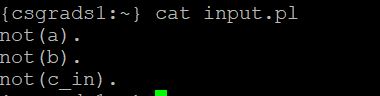
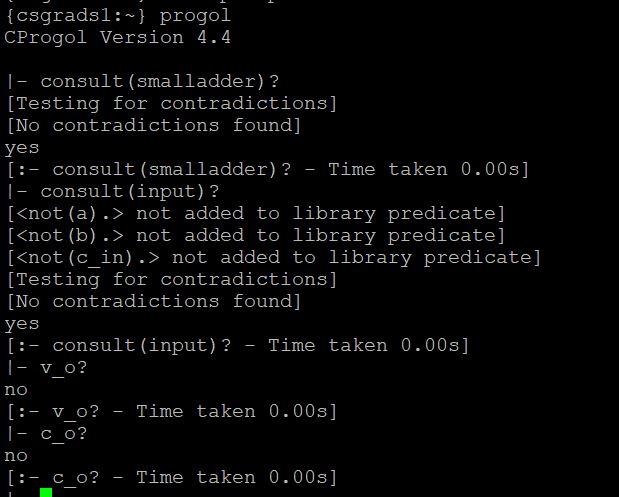


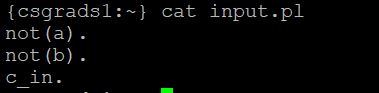
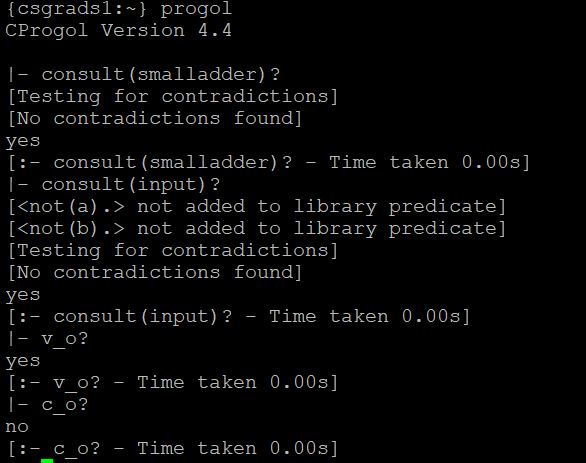
The truth table is verified and the results are as per the table.

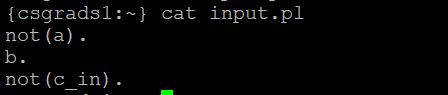
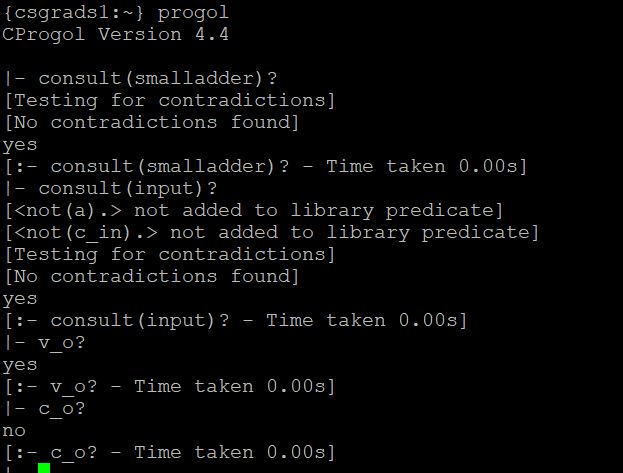
b)

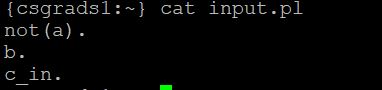
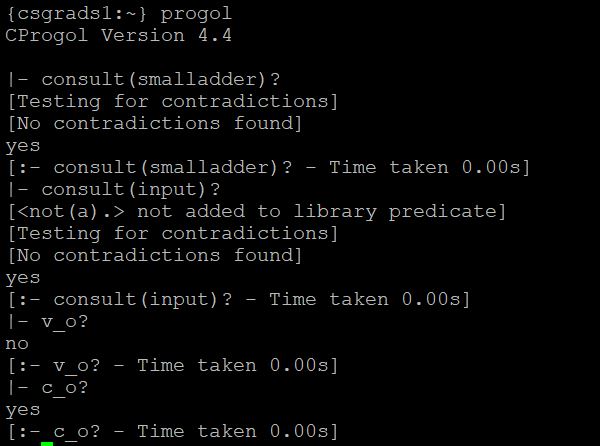
Yes. We can simplify further. As there are 4 output predicates (v\_o,c\_o,not\_v\_o and not\_c\_out), we can cut iyt down to 2 by only defining v\_o and c\_o. We use a meta-logical predicate called not() for the definitions not\_v\_o as not(v\_o) and not\_c\_o as not(c\_o)

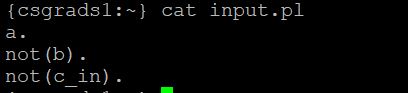
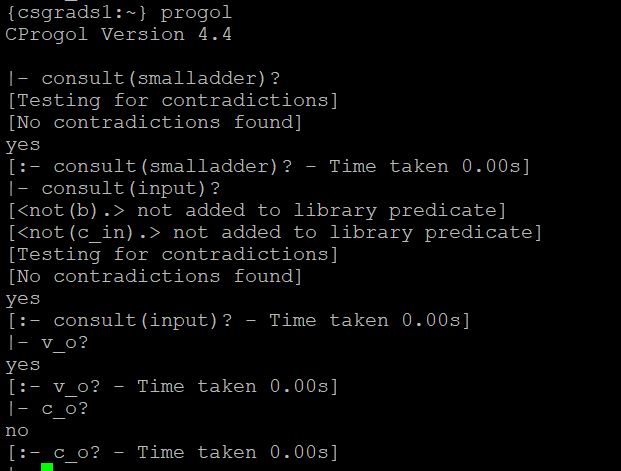
**Exercise 5:**

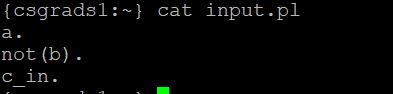
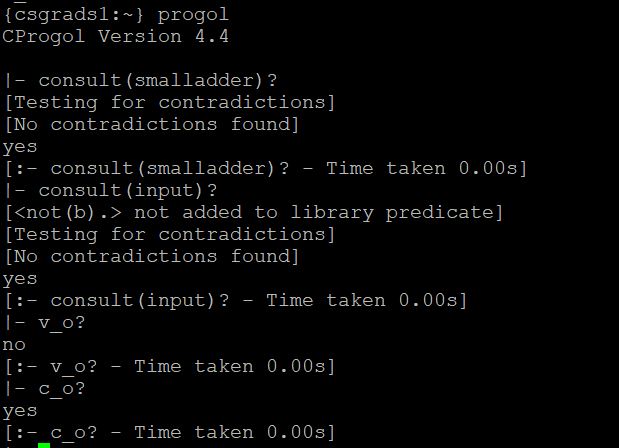
 

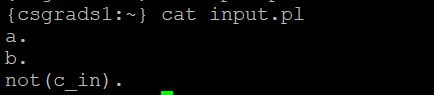
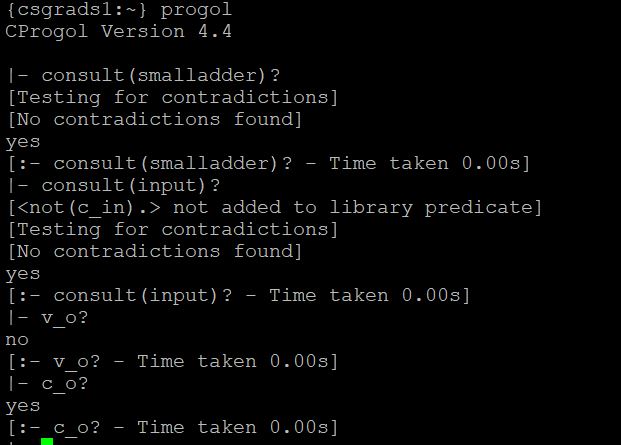
 

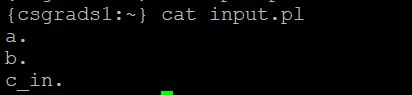
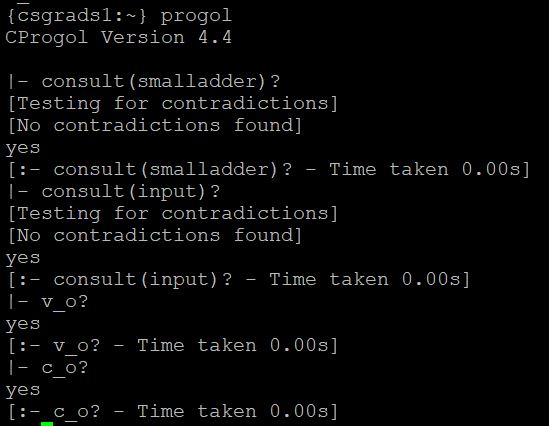
 

**Exercise 6:**

a) x in S1 & S2 iff x in S1 and x in S2

Definite clause:

in\_inter(X, S1, S2) :- elem(X, S1), elem(X, S2).

b)<x,y> in S1 x S2 iff x in S1 and y in S2

Definite clause:

in\_binary(<X,Y>, S1, S2) :- elem(X, S1), elem(Y, S2).

c) x in S1 \ S2 iff x in S1 and x not in S2

Definite clause:

in\_diff(X, S1, S2) :- elem(X, S1), not(elem(Y, S2)).

**Exercise 7:**

a) less\_than5({<x,y> | x in N5, y in N5, and x<y)--Given

Extensional definition:

less\_than5(0,1).

less\_than5(0,2).

less\_than5(0,3).

less\_than5(0,4).

less\_than5(1,2).

less\_than5(1,3).

less\_than5(1,4).

less\_than5(2,3).

less\_than5(2,4).

less\_than5(3,4).

b)

lt\_AxB(0,1).

lt\_AxB(0,2).

lt\_AxB(0,3).

lt\_AxB(0,4).

lt\_AxB(1,2).

lt\_AxB(1,3).

lt\_AxB(1,4).

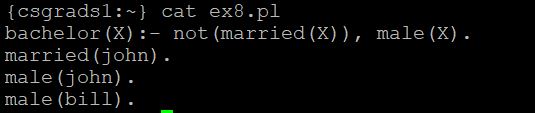
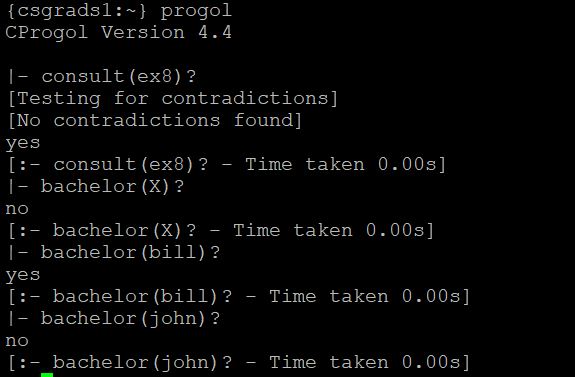
lt\_AxB(2,3).

lt\_AxB(2,4).

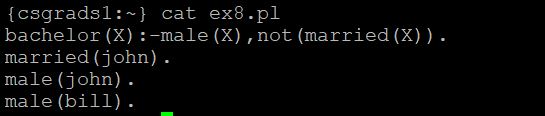
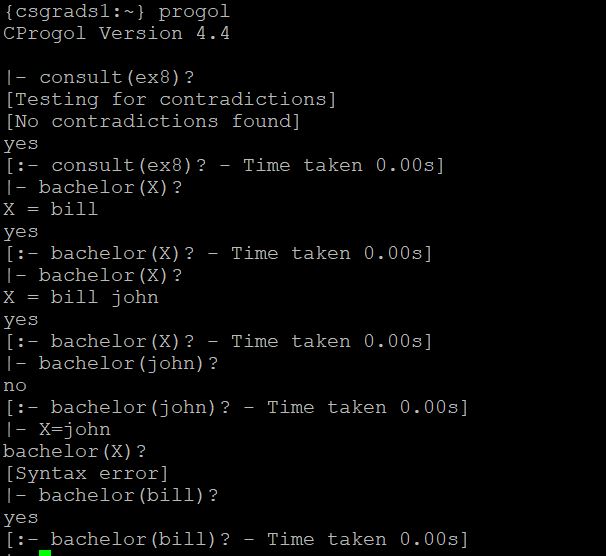
lt\_AxB(3,4).

**Exercise 8:**

a,b)

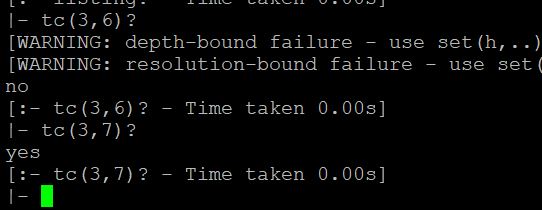
 

c)

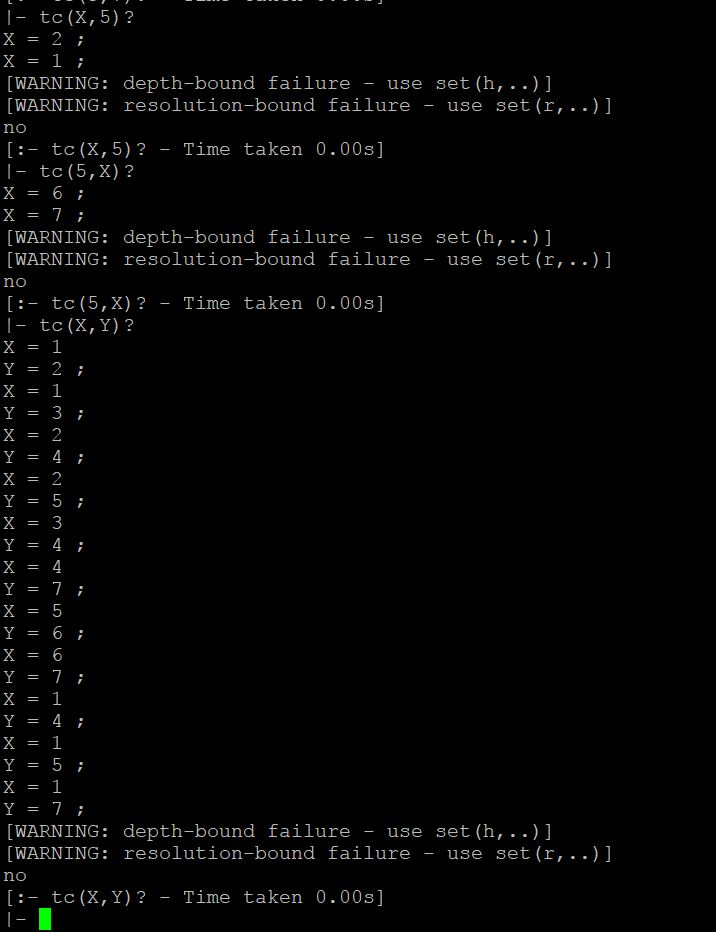
 

**Exercise 9:**

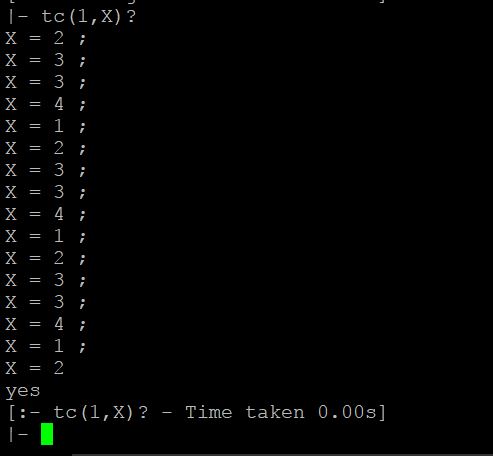
a)



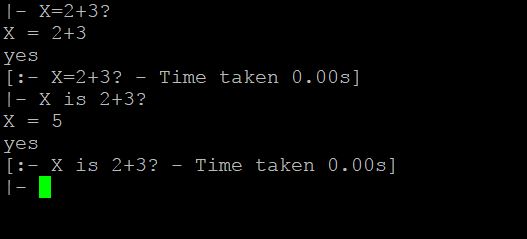
b)



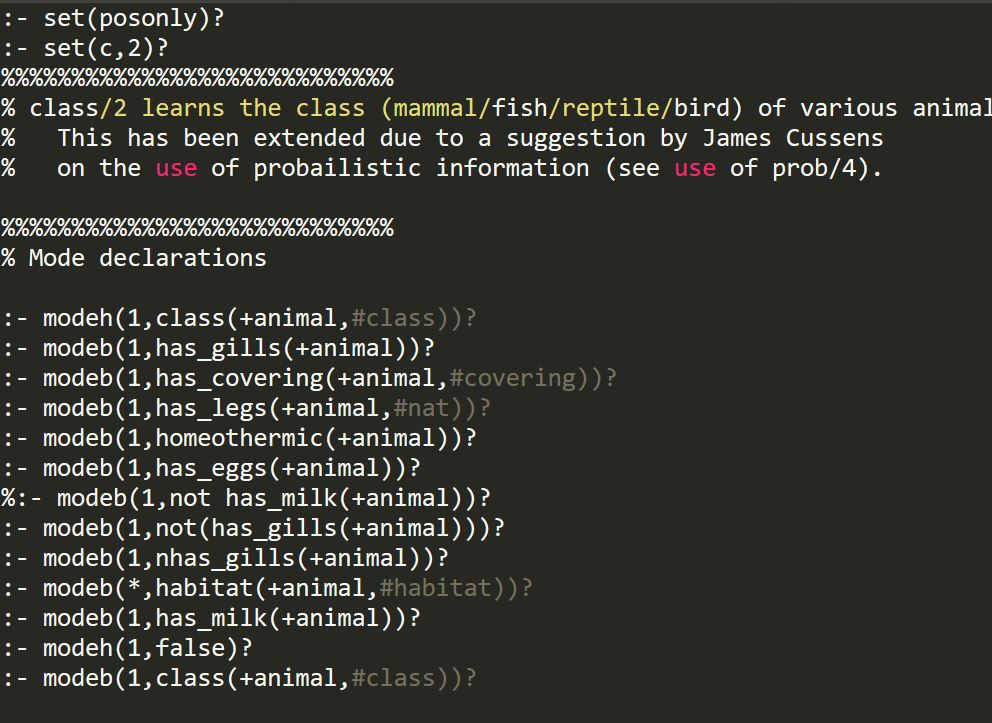
c)



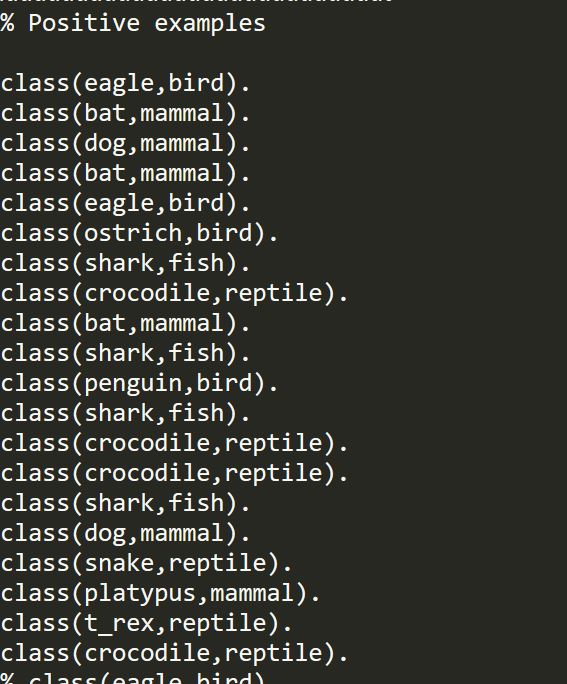
**Exercise 10:**



**Exercise 11:**







**Exercise 12:**

Predicates are: Here X is type of animal

has\_gills(X)

has\_covering(X)

has\_legs(X)

homeothermic(X)

has\_eggs(X)

nhas\_gills(X)

habitat(X)

has\_milk(X)

**Exercise 13:**

class(X):- member(X,[mammal,fish,reptile,bird]).

It is a valid type definition. First argument is meant to be variable i.e the first argument is an output variable and the second is an input variables.

**Exercise 14:**

a) Clause isn't allowed. Because, in class(+animal,#class), # indicates we must have constants i.e(B). But has\_milk takes variable of type animal.

b) Clause allowed. Since , has\_milk takes variable of type animal which is A.

c) Clause is allowed. Because has\_milk(platypus) is true as platypus is mammal and for any mammal.

d) Clause is allowed. Because has\_milk(platypus) is true and platypus is mammal.

e) Clause isn't allowed. Because all mammals won't be having milk.

**Exercise 15:**

Recall number for habitat/2 is \*, Because for given input, predicate habitat succeeds more than once.(or finite times)

Eg: For input bat, habitat successeds multiple times i.e air , caves

For input crocodile, habitat successeds multiple times i.e land , water

**Exercise 16:**

a) :- modeh(1,mult(+num,+num,-num)).

:- modeb(1,dec(+num,-num)).

:- modeb(1,plus(+num,+num,-num)).

:- modeb(1,nat(+num)).

b) :- modeh(1,n\_choose\_m((+num,+num,-num)).

:- modeb(1,dec(+num,-num)).

:- modeb(1,multiply(+num,+num,-num)).

:- modeb(1,divide(+num,+num,-num)).

:- modeb(1,nat(+num)).