Programming Assignment 2 : PROGOL

CS 6364

Submitted by:

Aashaar Panchalan – adp170630

Manish Biyani – mxb172930

Ex. 1:

```
aashaar@aashaar-VirtualBox:~/aashaar_data/PROGOL/source$ ./progol
CProgol Version 4.4

|-
```

Ex. 2:

```
|- help?
The following system predicates are available:
  !                , /2              -> /2             . /2
  ; /2             < /2              = /2              =.. /2
  =< /2            == /2             > /2              >= /2
  @< /2            @=< /2            @> /2             @>= /2
  \+ /1            \= /2             \== /2            advise/1
  aleave/1         any/1             arg/3             asserta/1
  assertz/1        bagof/3           chisq/4           clause/2
  clause/3         commutative/1     commutatives      constant/1
  constraint/1     constraints       consult/1         determination/2
  edit/1           element/2         fixedseed         float/1
  functor/3        generalise/1      halt/1            help
  help/1           hypothesis/3      in/2              int/1
  is/2             label/1           label/2           layer/1
  leave/1          length/2          listing           listing/1
  modeb/2          modeh/2           modes             name/2
  nat/1            nl                normal/2          normal/3
  nospy            not/1             notrace           number/1
  op/3             ops               optoggle          otherwise
  permute/1        prune             prune1/2          quit
  randomseed       read/1            read/2            read1/1
  reconsult/1      record/2          reduce/1          repeat
  retract/1        retract/2         sample/3          see/1
  seen             set/1             set/2             setof/3
  settings         solving           sort/2            spies
  spy/1            stats             system/1          tab/1
  tell/1           test/1            test/2            told
  trace            true              uniform/3         unset/1
  user_predicate/1 var/1             vassert/1         vretract/1
  write/1          writev/1
Help for system predicates using help(Predicate/Arity)
yes
[:- help? - Time taken 0.00s]
|-
```

Ex. 3:

Please refer the file ‘adder.pl’ in the codes folder, for the code.

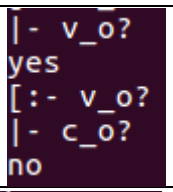
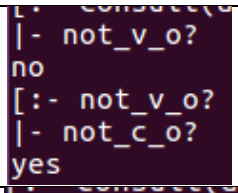
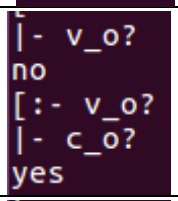
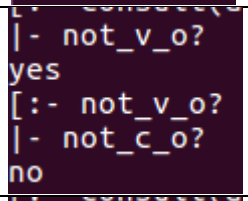
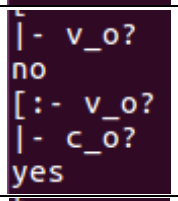
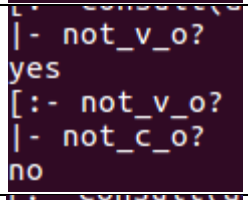
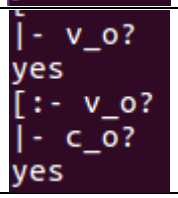
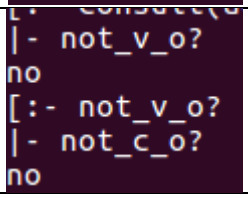
Here’s the screenshot for input state : a, not_b, not_c_in:

```
|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- listing?
The following user predicates are defined:
  a         c_o       not_b     not_c_in not_c_o  not_v_o  v_o
[Total number of clauses = 19]
yes
[:- listing? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|-
```

Ex. 4:

a)

A	B	Cin	Vo	Co	Progol Screenshot
0	0	0	0	0	`|- v_o?` `no` `[:- v_o?` `|- c_o?` `no` `|- not_v_o?` `yes` `[:- not_v_o?` `|- not_c_o?` `yes`
0	0	1	1	0	`|- v_o?` `yes` `[:- v_o?` `|- c_o?` `no` `|- not_v_o?` `no` `[:- not_v_o?` `|- not_c_o?` `yes`
0	1	0	1	0	`|- v_o?` `yes` `[:- v_o?` `|- c_o?` `no` `|- not_v_o?` `no` `[:- not_v_o?` `|- not_c_o?` `yes`
0	1	1	0	1	`|- v_o?` `no` `[:- v_o?` `|- c_o?` `yes` `|- not_v_o?` `yes` `[:- not_v_o?` `|- not_c_o?` `no`

1	0	0	1	0				
1	0	1	0	1				
1	1	0	0	1				
1	1	1	1	1				

From the truth-table, we have verified that Progol gives correct answers for all the output propositions.

b)

Yes, the adder circuit can be represented by fewer clauses by eliminating the negation of the two outputs. If we have the propositions for v_o and c_o we don't need to maintain not_v_o and not_c_o. if required, they can simply be obtained by negating the v_o & c_o respt. Thus we can eliminate 8 clauses and the same adder circuit can be formed using only 8 clauses – 4 each for v_o and c_o.

Ex. 5:

Please refer the file ‘adder2.pl’ in the codes folder, for the code.

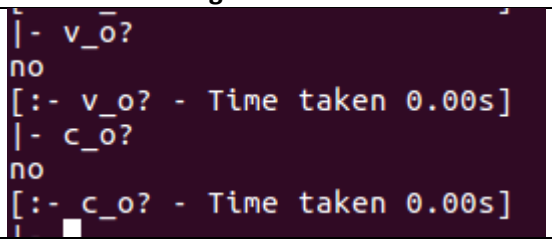
Below is the screenshot for input state : a, c (since we require not(b) to be true, we haven’t specified ‘b’ in the file so that by closed world assumption it becomes true.):

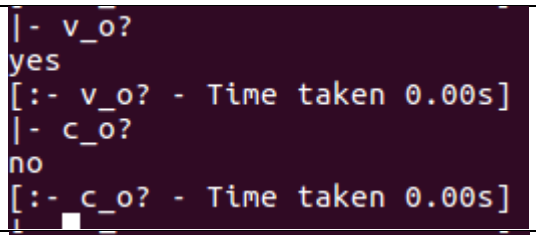
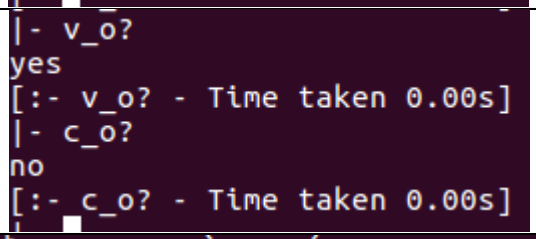
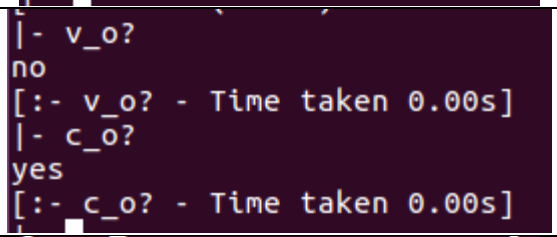
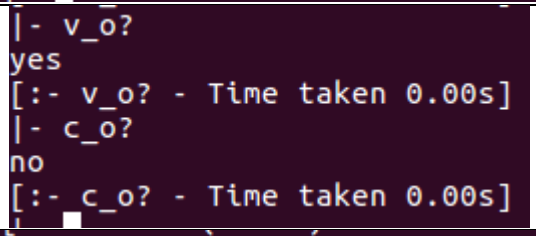
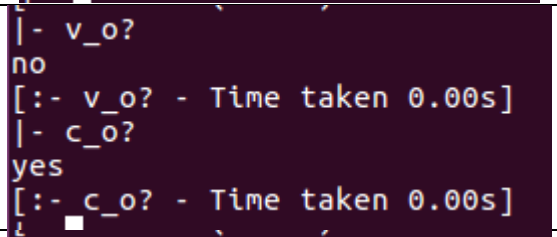
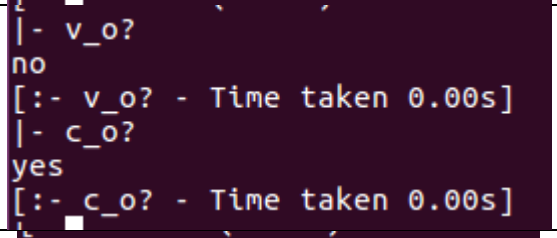
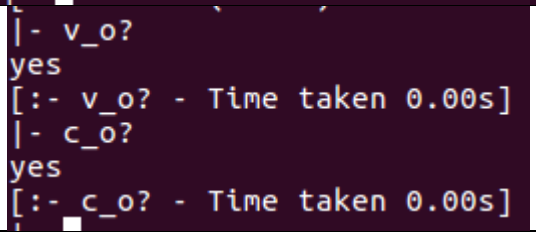
```
CProgol Version 4.4

|- consult(adder2)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder2)? - Time taken 0.00s]
|- listing?
The following user predicates are defined:
  a    c_in c_o  v_o
[Total number of clauses = 10]
yes
[:- listing? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- not(v_o)?
yes
[:- not(v_o)? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
|- not(c_o)?
no
[:- not(c_o)? - Time taken 0.00s]
|-
```

From the output in the above screenshot, we can see that not_v_o and not_c_o can be derived from the the v_o and c_o respt., simply by negating them due to the closed world assumption.

b)

A	B	Cin	Vo	Co	Progol Screenshot
0	0	0	0	0	

0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	

From the truth-table, we have verified that Progol gives correct answers for all the output propositions.

Ex. 6:

a) in_and(X,S1,S2):-
elem(X,S1), elem(X,S2).

b) in_mult(X,Y,S1,S2):-
elem(X,S1), elem(Y,S2).

c) in_div(X,S1,S2):-
elem(X,S1), not(elem(X,S2)).

Ex. 7:

a) less_than_5(0,1).
less_than_5(0,2).
less_than_5(0,3).
less_than_5(0,4).
less_than_5(1,2).
less_than_5(1,3).
less_than_5(1,4).
less_than_5(2,3).
less_than_5(2,4).
less_than_5(3,4).

b) in_mult(X,Y,A,B):-
elem(X,A),elem(X,B),less_than_5(X,Y).

Ex. 8 :

a) bachelor(X) returns NO.
b) bachelor(bill) returns YES and bachelor(john) returns NO.
Please refer below screenshot for the above two answers.

```
|- bachelor(X)?
no
[:- bachelor(X)? - Time taken 0.00s]
|- bachelor(bill)?
yes
[:- bachelor(bill)? - Time taken 0.00s]
|- bachelor(john)?
no
[:- bachelor(john)? - Time taken 0.00s]
```

c) bachelor(X) gives
X=Bill
Yes.
bachelor(bill) returns YES and bachelor(john) returns NO.
Please refer below screenshot for the above two answers.

```
|- bachelor(X)?
X = bill
yes
[:- bachelor(X)? - Time taken 0.00s]
|- bachelor(bill)?
yes
[:- bachelor(bill)? - Time taken 0.00s]
|- bachelor(john)?
no
[:- bachelor(john)? - Time taken 0.00s]
```

Ex. 9:

We added all the predicates to a file named tc.pl (available in codes folder) and changed the definition for each question.

a) **Definition 1:**

tc(3,6)- No

tc(3,7) – Yes

```
|- tc(3,6)?
no
[:- tc(3,6)? - Time taken 0.00s]
|- tc(3,7)?
yes
```

Definition 2:

tc(3,6)- No

tc(3,7) – Yes

```
|- tc(3,6)?
[WARNING: depth-bound failure - use set(h,..)]
no
[:- tc(3,6)? - Time taken 0.00s]
|- tc(3,7)?
yes
[:- tc(3,7)? - Time taken 0.00s]
```

Definition 3:

tc(3,6)- No

tc(3,7) – Yes

```
|- tc(3,6)?
[WARNING: depth-bound failure - use set(h,..)]
no
[:- tc(3,6)? - Time taken 0.00s]
|- tc(3,7)?
[WARNING: depth-bound failure - use set(h,..)]
yes
```

b) **Definition 1:**

tc(X,5)

X = 2 ;

X = 1 ;

```
|- tc(X,5)?
X = 2
yes
[:- tc(X,5)? - Time taken 0.00s]
|- tc(X,5)?
X = 2 ;
X = 1 ;
no
```

tc(5,X)

X = 6 ;

X = 7 ;

```
|- tc(5,X)?
X = 6 ;
X = 7 ;
no
```

tc(X,Y)

X = 1

Y = 2 ;

X = 1

Y = 3 ;

X = 2

Y = 4 ;

X = 2

Y = 5 ;

X = 3

Y = 4 ;

X = 4

Y = 7 ;

X = 5

Y = 6 ;

X = 6

Y = 7 ;

X = 1

Y = 4 ;

X = 1

Y = 5 ;

X = 1

Y = 7 ;

X = 1

Y = 6 ;

X = 1

Y = 7 ;

X = 1

Y = 4 ;

X = 1

Y = 7 ;

X = 2

Y = 7 ;

X = 2

Y = 6 ;

X = 2

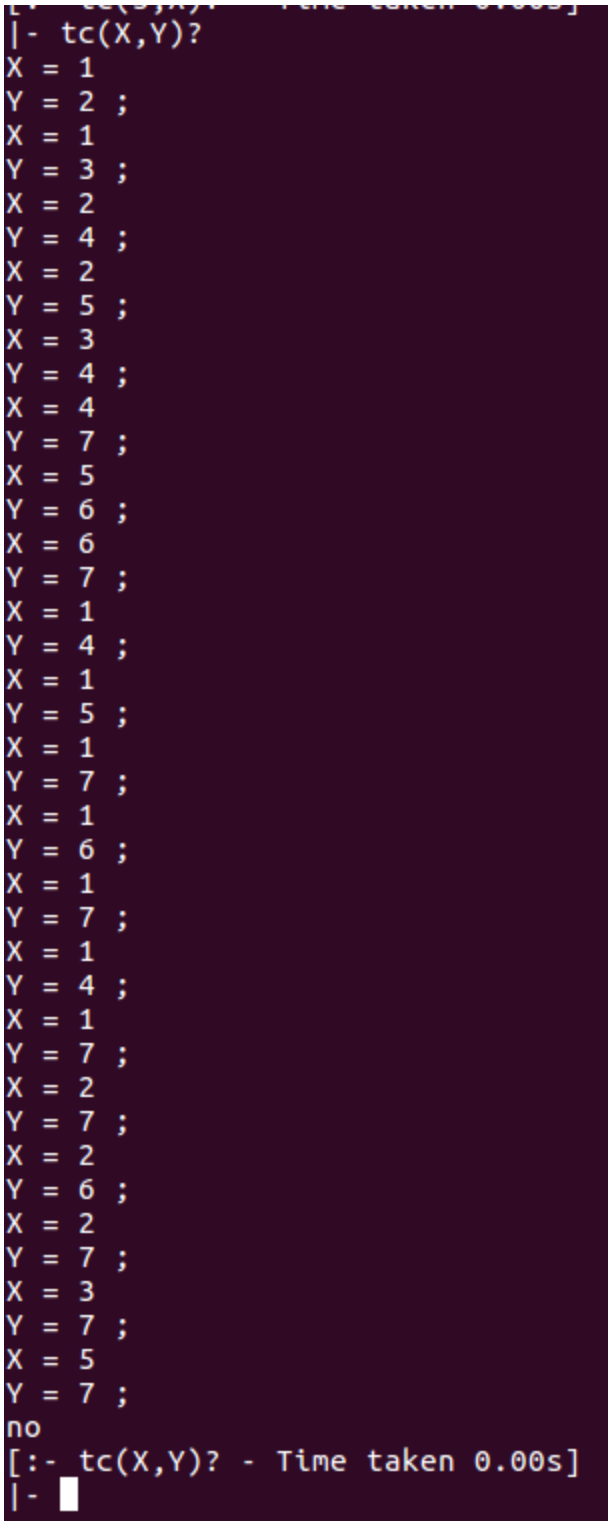
Y = 7 ;

X = 3

Y = 7 ;

X = 5

Y = 7 ;


|- tc(X,Y)?
X = 1
Y = 2 ;
X = 1
Y = 3 ;
X = 2
Y = 4 ;
X = 2
Y = 5 ;
X = 3
Y = 4 ;
X = 4
Y = 7 ;
X = 5
Y = 6 ;
X = 6
Y = 7 ;
X = 1
Y = 4 ;
X = 1
Y = 5 ;
X = 1
Y = 7 ;
X = 1
Y = 6 ;
X = 1
Y = 7 ;
X = 1
Y = 4 ;
X = 1
Y = 7 ;
X = 2
Y = 7 ;
X = 2
Y = 6 ;
X = 2
Y = 7 ;
X = 3
Y = 7 ;
X = 5
Y = 7 ;
no
[:- tc(X,Y)? - Time taken 0.00s]
|-

(Ex.9b conted …)

Definition 2:

tc(X,5)

X = 2;

X = 1;

```
[:- tc(3,7)? - Time taken 0.00s]
|- tc(X,5)?
X = 2 ;
X = 1 ;
[WARNING: resolution-bound failure - use set(r,..)]
no
```

tc(5,X)

X = 6;

X =7;

```
[:- tc(X,5)? - Time taken 0.00s]
|- tc(5,X)?
X = 6 ;
X = 7 ;
[WARNING: depth-bound failure - use set(h,..)]
no
```

tc(X,Y)

X = 1

Y = 2 ;

X = 1

Y = 3 ;

X = 2

Y = 4 ;

X = 2

Y = 5 ;

X = 3

Y = 4 ;

X = 4

Y = 7 ;

X = 5

Y = 6 ;

X = 6

Y = 7 ;

X = 1

Y = 4 ;

X = 1

Y = 5 ;

X = 1

Y = 4 ;

X = 2

Y = 7 ;

X = 2

Y = 6 ;

X = 3

Y = 7 ;

X = 5

Y = 7 ;

X = 1

Y = 7 ;

X = 1

Y = 6 ;

X = 1

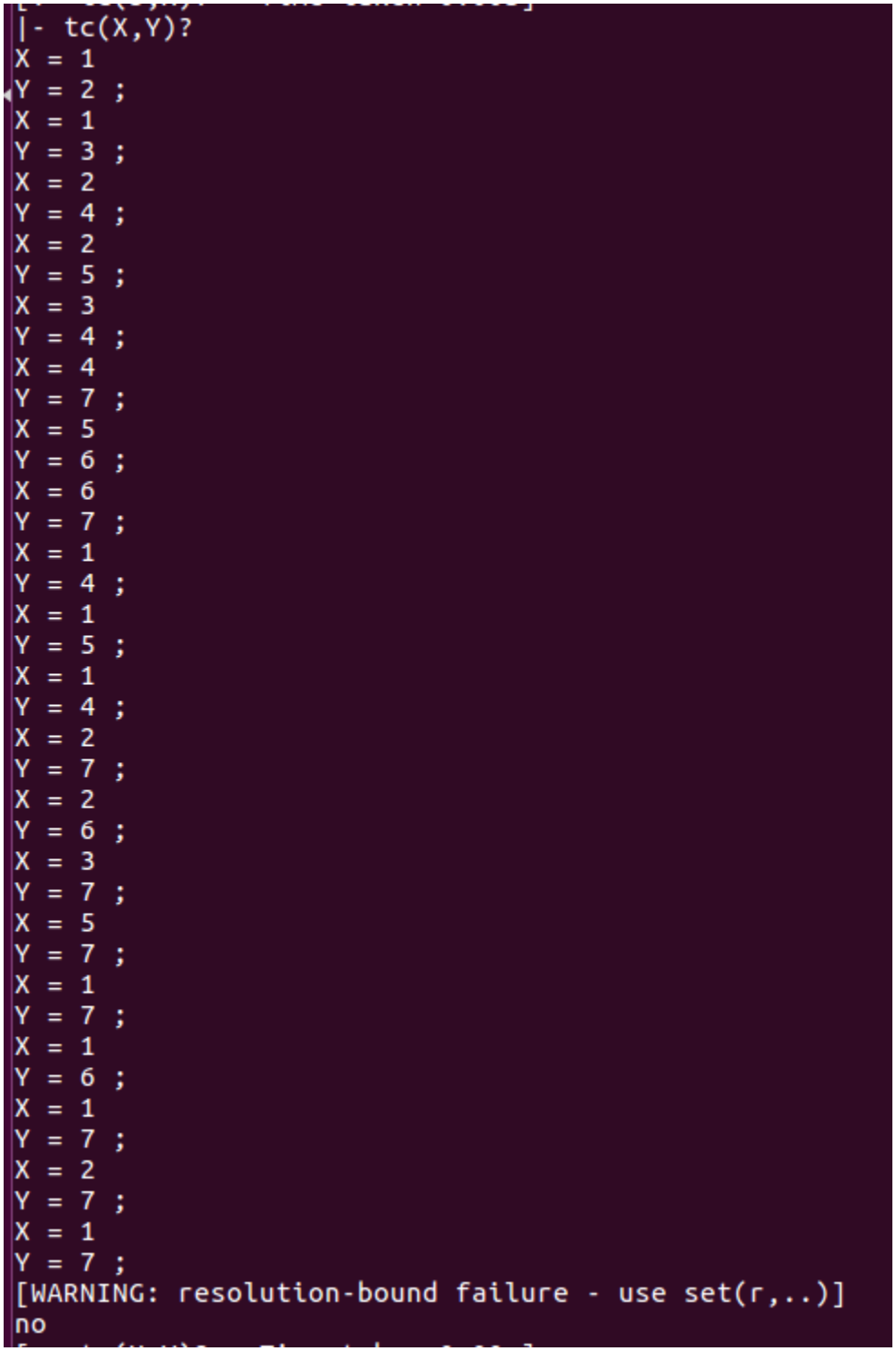
Y = 7 ;

X = 2

Y = 7 ;

X = 1

Y = 7 ;


|- tc(X,Y)?
X = 1
Y = 2 ;
X = 1
Y = 3 ;
X = 2
Y = 4 ;
X = 2
Y = 5 ;
X = 3
Y = 4 ;
X = 4
Y = 7 ;
X = 5
Y = 6 ;
X = 6
Y = 7 ;
X = 1
Y = 4 ;
X = 1
Y = 5 ;
X = 1
Y = 4 ;
X = 2
Y = 7 ;
X = 2
Y = 6 ;
X = 3
Y = 7 ;
X = 5
Y = 7 ;
X = 1
Y = 7 ;
X = 1
Y = 6 ;
X = 1
Y = 7 ;
X = 2
Y = 7 ;
X = 1
Y = 7 ;
[WARNING: resolution-bound failure - use set(r,..)]
no

(Ex.9b conted …)

Definition 3:

tc(X,5) – None

```
tc(X,5)?
[WARNING: depth-bound failure - use set(h,..)]
[WARNING: resolution-bound failure - use set(r,..)]
no
```

tc(5,X)

X = 6;

X =7;

```
|- tc(5,X)?
[WARNING: depth-bound failure - use set(h,..)]
X = 7 ;
X = 6 ;
no
[:- tc(5,X)? - Time taken 0.00s]
```

tc(X,Y) – None

```
|- tc(X,Y)?
[WARNING: depth-bound failure - use set(h,..)]
[WARNING: resolution-bound failure - use set(r,..)]
no
```

c)

X = 2 ;

X = 3 ;

X = 3 ;

X = 4 ;

X = 1 ;

X = 2 ;

X = 3 ;

X = 3 ;

X = 4 ;

X = 1 ;

```
|- tc(1,X)?
X = 2 ;
X = 3 ;
X = 3 ;
X = 4 ;
X = 1 ;
X = 2 ;
X = 3 ;
X = 3 ;
X = 4 ;
X = 1 ;
```

Ex. 10:

a) X= 2+3? - Yes

b) X is 2+3?

X = 5?

Yes

```
|- X = 2 + 3?
X = 2+3
yes
[:- X=2+3? - Time taken 0.00s]
|- X is 2 + 3?
X = 5
yes
[:- X is 2+3? - Time taken 0.00s]
```

Ex. 11:

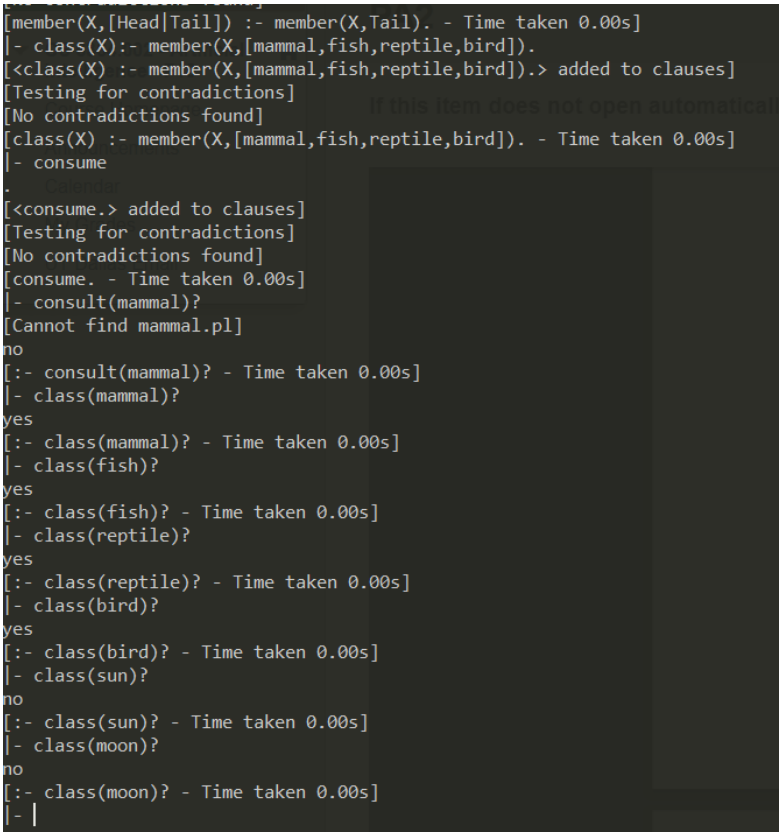
```
|- consult(animals)?
[Noise has been set to 100%]
[Example inflation has been set to 400%]
[The posonly flag has been turned ON]
[:- set(posonly)? - Time taken 0.00s]
[:- set(c,2)? - Time taken 0.00s]
[:- modeh(1,class(+animal,#class))? - Time taken 0.00s]
[:- modeb(1,has_gills(+animal))? - Time taken 0.00s]
[:- modeb(1,has_covering(+animal,#covering))? - Time taken 0.00s]
[:- modeb(1,has_legs(+animal,#nat))? - Time taken 0.00s]
[:- modeb(1,homeothermic(+animal))? - Time taken 0.00s]
[:- modeb(1,has_eggs(+animal))? - Time taken 0.00s]
[:- modeb(1,not(has_gills(+animal)))? - Time taken 0.00s]
[:- modeb(1,nhas_gills(+animal))? - Time taken 0.00s]
[:- modeb(100,habitat(+animal,#habitat))? - Time taken 0.00s]
[:- modeb(1,has_milk(+animal))? - Time taken 0.00s]
[:- modeh(1,false)? - Time taken 0.00s]
[:- modeb(1,class(+animal,#class))? - Time taken 0.00s]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(animals)? - Time taken 0.00s]
|-
```

Ex. 12:

has_gills/1	args: animal
has_covering/2	args: animal, covering
has_legs/2	args: animal, count
homeothermic/1	args: animal
has_eggs/1	args: animal
nhas_gills/1	args: animal
habitat/2	args: animal, habitat
has_milk/1	args: animal
class/2	args: animal, class

Ex. 13:

Yes, it is legal.
X takes the value given when we query the data. The definition will check if the given X is a member of the list [mammal, fish, reptile, bird]. If the value returns True, Class(X) is also true else it is false.



Ex: 14

(a) class(A,B):- has_milk(B).
Not allowed. The first argument in class/2 should have the same argument in has_milk/1
(b) class(A,B):- has_milk(A).
Not allowed. 2nd argument in class/2 should be a constant.
(c) class(A, mammal):-has_milk(platypus).
Not allowed. The argument in has_milk/1 should not be a constant in the definition. The argument in has_milk/1 should be the same as the first argument in class/2.
(d) class(platypus, mammal):- has_milk(platypus).
Not allowed. The argument in has_milk/1 and first argument in class/2 should not be a constant.
(e) class(A, mammal):- has_milk(A).
Allowed. The above example satisfies the language constraints..

Ex: 15

For Eg: habitat(crocodile, land)succeeds. Similarly habitat(crocodile, water) also succeeds.

Habitat is a predicate which can return finite number of output values and so we write habitat/2 *

```
|- habitat(crocodile,X)?
X = water ;
X = land ;
no
[:- habitat(crocodile,X)? - Time taken 0.00s]
|- habitat(cow,X)?
no
[:- habitat(cow,X)? - Time taken 0.00s]
|- habitat(dog,X);?

.
[<habitat(dog,X); ?.> not added to library predicate]
[habitat(dog,X); ?. - Time taken 0.00s]
|- habitat(dog,X)?
X = land ;
no
[:- habitat(dog,X)? - Time taken 0.00s]
|-
```

Ex: 16

a)

```
|-  :- modeh(1,mult(+X,+Y,-Z))?
no
[:- modeh(1,mult(+X,+Y,-Z))? - Time taken 0.00s]
|- :- modeb(1,dec(+X,-D))?
no
[:- modeb(1,dec(+X,-D))? - Time taken 0.00s]
|- :- modeb(1,mult(+X,+Y,-Z))?
no
[:- modeb(1,mult(+X,+Y,-Z))? - Time taken 0.00s]
|-  :- modeb(1,sum(+X,+Y,-Z))?
no
[:- modeb(1,sum(+X,+Y,-Z))? - Time taken 0.00s]
|-
```

b)

```
|- :-modeh(1,n_choose_m(+A,+B,-C))?
no
[:- modeh(1,n_choose_m(+A,+B,-C))? - Time taken 0.00s]
|- :-modeb(1,dec(+B,-D))?
no
[:- modeb(1,dec(+B,-D))? - Time taken 0.00s]
|- :-modeb(1,n_choose_m(+E,+D,-F))?
no
[:- modeb(1,n_choose_m(+E,+D,-F))? - Time taken 0.00s]
|- :-modeb(1,multiply(+F,+A,-G))?
no
[:- modeb(1,multiply(+F,+A,-G))? - Time taken 0.00s]
|- modeb^[[D^[[D^Xc
[Syntax error]
|- :-modeb(1,divide(+G,+B,-C))?
no
[:- modeb(1,divide(+G,+B,-C))? - Time taken 0.00s]
|- n_choose_m(A,B,C):-  dec(B,D),  dec(A,E),  n_choose_m(E,D,F),  multiply(F,A,G),  divide(G,B,C).
[<n_choose_m(A,B,C) :- dec(B,D), dec(A,E), n_choose_m(E,D,F), multiply(F,
       A,G), divide(G,B,C).> added to clauses]
[Testing for contradictions]
[No contradictions found]
[n_choose_m(A,B,C) :- dec(B,D), dec(A,E), n_choose_m(E,D,F), multiply(F,A,G), divide(G,B,C). - Time taken 0.00s]
|-
```
