CS 6364.002 ARTIFICIAL INTELLIGENCE
Programming Assignment – 2
October 24, 2017

Dileep Gudena (dxg161730)
Gokul Surendra (gxs161530)

Exercise 1:

```
{csgrads1:~} progol
CProgol Version 4.4

|-
```

Exercise 2:

```
{csgrads1:~} progol
CProgol Version 4.4

|- 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]
|-
{csgrads1:~}
```

Exercise 3:

```
{csgrads1:~} cat adder.pl
v_o :- not_a, not_b, c_in.
v_o :- not_a, b, not_c_in.
v_o :- a, not_b, not_c_in.
v_o :- a, b, c_in.
not_v_o :- not_a, not_b, not_c_in.
not_v_o :- not_a, b, c_in.
not_v_o :- a, not_b, c_in.
not_v_o :- a, b, not_c_in.
c_o :- not_a, b, c_in.
c_o :- a, not_b, c_in.
c_o :- a, b, not_c_in.
c_o :- a, b, c_in.
not_c_o :- not_a, not_b, not_c_in.
not_c_o :- not_a, not_b, c_in.
not_c_o :-not_a, b, not_c_in.
not_c_o :- a, not_b, not_c_in.
```

```
{csgrads1:~} cat input.pl
a.
not_b.
not_c_in.
```

```
{csgrads1:~} vi adder.pl
{csgrads1:~} vi input.pl
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)
?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - 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]
|- listing(not_c_o)?

not_c_o :- not_a, not_b, not_c_in.
not_c_o :- not_a, not_b, c_in.
not_c_o :- not_a, b, not_c_in.
not_c_o :- a, not_b, not_c_in.

[Total number of clauses = 4]
yes
[:- listing(not_c_o)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
```

Exercise 4:

```
{csgrads1:~} cat input.pl
not_a.
not_b.
not_c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)
?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
|- not_v_o?
yes
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
yes
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not_a.
not_b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
|- not_v_o?
no
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
yes
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not_a.
b.
not_c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
|- not_v_o?
no
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
yes
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not_a.
b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
|- not_v_o?
yes
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
no
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
not_b.
not_c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
|- not_v_o?
no
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
yes
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
not_b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
|- not_v_o?
yes
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
no
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
b.
not_c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
|- not_v_o?
yes
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
no
[:- not_c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(adder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(adder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
|- not_v_o?
no
[:- not_v_o? - Time taken 0.00s]
|- not_c_o?
no
[:- not_c_o? - Time taken 0.00s]
```

A	B	Cin	Vo	Co
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

truth-table specification for this adder

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:

```
{csgrads1:~} cat input.pl
not(a).
not(b).
not(c_in).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(a).> not added to library predicate]
[<not(b).> not added to library predicate]
[<not(c_in).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not(a).
not(b).
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(a).> not added to library predicate]
[<not(b).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not(a).
b.
not(c_in).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(a).> not added to library predicate]
[<not(c_in).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
not(a).
b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(a).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
not(b).
not(c_in).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(b).> not added to library predicate]
[<not(c_in).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
no
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
not(b).
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(b).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
b.
not(c_in).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[<not(c_in).> not added to library predicate]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
no
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
```

```
{csgrads1:~} cat input.pl
a.
b.
c_in.
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(smalladder)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(smalladder)? - Time taken 0.00s]
|- consult(input)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(input)? - Time taken 0.00s]
|- v_o?
yes
[:- v_o? - Time taken 0.00s]
|- c_o?
yes
[:- c_o? - Time taken 0.00s]
```

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)

```
{csgrads1:~} cat ex8.pl
bachelor(X):- not(married(X)), male(X).
married(john).
male(john).
male(bill).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(ex8)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(ex8)? - Time taken 0.00s]
|- 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)

```
{csgrads1:~} cat ex8.pl
bachelor(X):-male(X),not(married(X)).
married(john).
male(john).
male(bill).
```

```
{csgrads1:~} progol
CProgol Version 4.4

|- consult(ex8)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(ex8)? - Time taken 0.00s]
|- bachelor(X)?
X = bill
yes
[:- bachelor(X)? - Time taken 0.00s]
|- bachelor(X)?
X = bill john
yes
[:- bachelor(X)? - Time taken 0.00s]
|- bachelor(john)?
no
[:- bachelor(john)? - Time taken 0.00s]
|- X=john
bachelor(X)?
[Syntax error]
|- bachelor(bill)?
yes
[:- bachelor(bill)? - Time taken 0.00s]
```

Exercise 9:

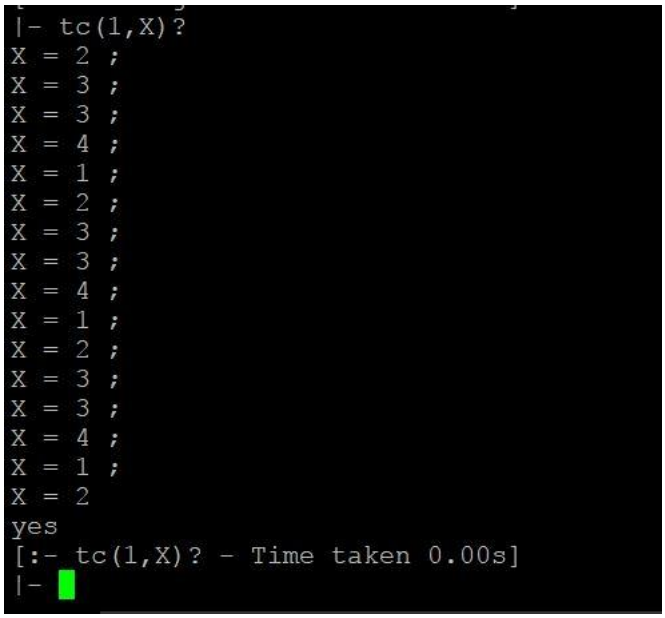
a)

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

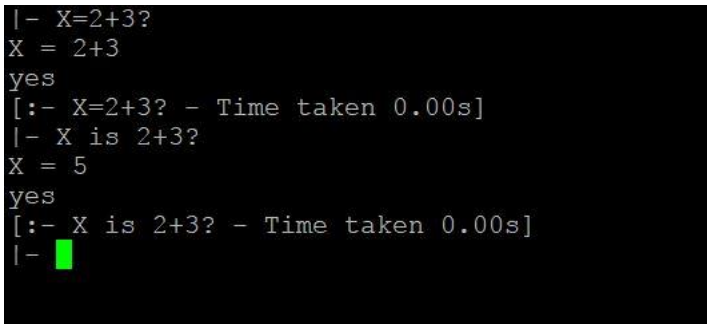
b)

```
|- tc(X,5)?
X = 2 ;
X = 1 ;
[WARNING: depth-bound failure - use set(h,..)]
[WARNING: resolution-bound failure - use set(r,..)]
no
[:- tc(X,5)? - Time taken 0.00s]
|- tc(5,X)?
X = 6 ;
X = 7 ;
[WARNING: depth-bound failure - use set(h,..)]
[WARNING: resolution-bound failure - use set(r,..)]
no
[:- tc(5,X)? - Time taken 0.00s]
|- 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 ;
[WARNING: depth-bound failure - use set(h,..)]
[WARNING: resolution-bound failure - use set(r,..)]
no
[:- tc(X,Y)? - Time taken 0.00s]
|-
```

c)



Exercise 10:



Exercise 11:

```
:- set(posonly)?
:- set(c,2)?
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% class/2 learns the class (mammal/fish/reptile/bird) of various animal
%   This has been extended due to a suggestion by James Cussens
%   on the use of probailistic information (see use of prob/4).

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mode declarations

:- modeh(1,class(+animal,#class))?
:- modeb(1,has_gills(+animal))?
:- modeb(1,has_covering(+animal,#covering))?
:- modeb(1,has_legs(+animal,#nat))?
:- modeb(1,homeothermic(+animal))?
:- modeb(1,has_eggs(+animal))?
%:- modeb(1,not has_milk(+animal))?
:- modeb(1,not(has_gills(+animal)))?
:- modeb(1,nhas_gills(+animal))?
:- modeb(*,habitat(+animal,#habitat))?
:- modeb(1,has_milk(+animal))?
:- modeh(1,false)?
:- modeb(1,class(+animal,#class))?
```

```
% Background knowledge

has_covering(dog,hair).
has_covering(dolphin,none).
has_covering(platypus,hair).
has_covering(bat,hair).
has_covering(trout,scales).
has_covering(herring,scales).
has_covering(shark,none).
has_covering(eel,none).
has_covering(lizard,scales).
has_covering(crocodile,scales).
has_covering(t_rex,scales).
has_covering(snake,scales).
has_covering(turtle,scales).
has_covering(eagle,feathers).
has_covering(ostrich,feathers).
has_covering(penguin,feathers).

has_legs(dog,4).
has_legs(dolphin,0).
has_legs(platypus,2).
has_legs(bat,2).
has_legs(trout,0).
has_legs(herring,0).
has_legs(shark,0).
has_legs(eel,0).
```

```
% Positive examples

class(eagle,bird).
class(bat,mammal).
class(dog,mammal).
class(bat,mammal).
class(eagle,bird).
class(ostrich,bird).
class(shark,fish).
class(crocodile,reptile).
class(bat,mammal).
class(shark,fish).
class(penguin,bird).
class(shark,fish).
class(crocodile,reptile).
class(crocodile,reptile).
class(shark,fish).
class(dog,mammal).
class(snake,reptile).
class(platypus,mammal).
class(t_rex,reptile).
class(crocodile,reptile).
% class(eagle,bird)
```

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)).
```