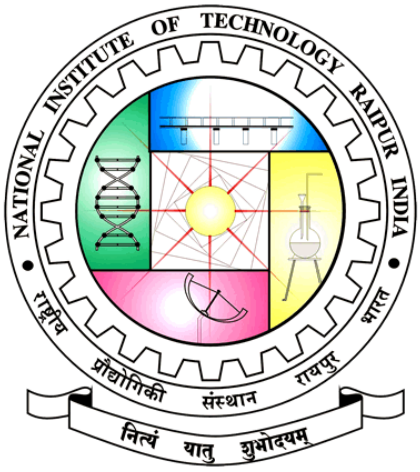
NATIONAL INSTITUTE OFTECHNOLOGY RAIPUR



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Artificial Intelligence & Expert Systems

Lab Course Code: CS20721(CS)

NAME – ANANY SHARMA **ROLL NO. –** 18115011

COURSE – B. TECH. **SEMESTER –** 7TH

Experiment: 17

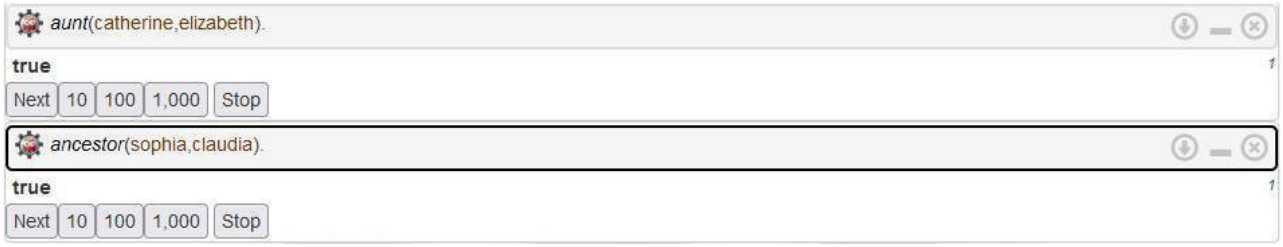
Aim: Write a prolog program to find the rules for parent, child, male, female, son, daughter, brother, sister, uncle, aunt, ancestor given the facts about father and wife only.

Program:

```
/* all the males */
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1). male(paul).
male(sam).

/* all the females */ female(catherine).
female(elizabeth). female(sophia).
female(claudia). female(fay).

/* parent child relationship */ parent(charles1,
james1). parent(elizabeth, james1).
parent(charles2, charles1). parent(catherine,
charles1). parent(james2, charles1).
parent(sophia, elizabeth). parent(george1,
sophia). parent(george1, sam).
parent(catherine, fay). parent(charles2, fay).
parent(james2, fay). parent(sophia, paul).
parent(elizabeth,claudia). parent(charles1,
claudia).

/* married relationship */ married(
james1, claudia). married( claudia,
james1). married(charles1, fay).
married(fay, charles1).
married(elizabeth, paul). married(paul,
elizabeth). married(sophia, sam).
married(sam, sophia).

/*rule for father */ father(Child, Dad) :- male(Dad),
parent(Child, Dad). /*rule for mother */
mother(Child, Mom) :- female(Mom), parent(Child,
Mom).
/*rule for daughter */ daughter(Child, Parent) :-
female(Child), parent(Child,Parent).
/* rule for son*/ son(Child, Parent) :- male(Child),
parent(Child,Parent).
```

```
/*rule for brother */
brother(Sibling, Bro) :- male(Bro), father(Sibling, Father), father(Bro, Father), Bro \= Sibling,

                                                    mother(Sibling, Mother), mother(Bro, Mother).
/*rule for sister */
sister(Sibling, Sis) :- female(Sis), father(Sibling, Father), father(Sis, Father), Sis \= Sibling,

                                                    mother(Sibling, Mother), mother(Sis, Mother).
/*rule for aunt */ aunt(Kid, Auntie) :- female(Auntie), parent(Kid, Parent), sister(Parent, Auntie). aunt(Kid,
Auntie) :- female(Auntie), parent(Kid, Person), brother(Person, Brother), married(Auntie, Brother).
/*rule for uncle */
uncle(Kid, UncleBuck) :- male(UncleBuck), parent(Kid, Parent), brother(Parent,
UncleBuck).
uncle(Kid, UncleBuck) :- male(UncleBuck), parent(Kid, Person), sister(Person, Sister),
married(UncleBuck, Sister). /*rule for ancestor*/
ancestor(Person, Ancestor) :- parent(Person, Ancestor). ancestor(Person, Ancestor) :- parent(Person,
Parent), ancestor(Parent, Ancestor).
```

Output:




father(charles1,james1).
true
Next 10 100 1,000 Stop
mother(james2,fay).
true
son(james2,charles1).
true
Next 10 100 1,000 Stop
daughter(elizabeth,james1).
true
Next 10 100 1,000 Stop

Experiment ; 18

Aim: Write a program to find the length last element of a given list.

Program:

```
length(X):- util(X,Count), write("The length of list
   is:"),write(Count). util([],X):-
   X=0.
util([_|T],Count):- util(T,Temp), Count is Temp+1.


last([X]):write("The last element of list is: "),write(X).
   last([_|T]):last(T).
```

Experiment:19

Aim: Write a program to delete the first occurrence and also all occurrences of a particular element in a given list.

Program:

```
deleteAll(_,[],[]). deleteAll(X,[H|T],[H|NT]):-

H\=X,deleteAll(X,T,NT). deleteAll(X,[X|T],NT):-

deleteAll(X,T,NT).
```

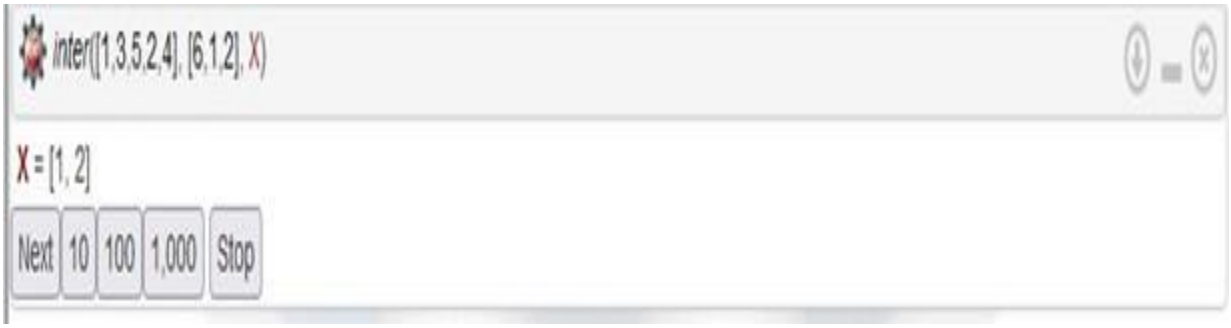
Output:

EXPERIMENT : 20

Aim: Write a program to find union and intersection of two given sets represented as lists.

Program:

```
union([X|Y],Z,W) :- member(X,Z), union(Y,Z,W).
union([X|Y],Z,[X|W]) :- \+ member(X,Z),
union(Y,Z,W). union([],Z,Z).
```



Output:

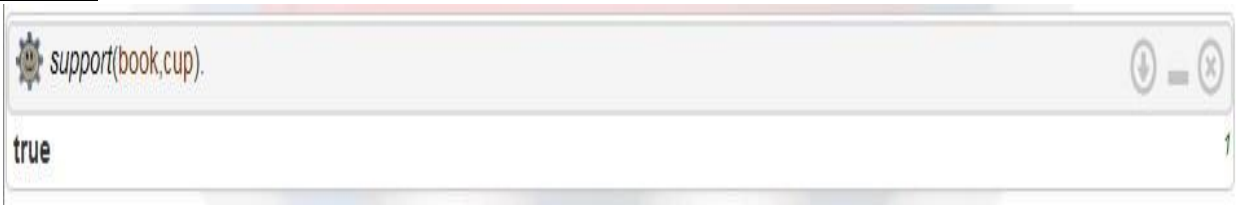
Experiment:22

Aim: Write a program given the knowledge base, If x is on the top of y, y supports x. If x is above y and they are touching each other, x is on top of y. A cup is above a book. The cup is touching that book. Convert the following into wffs, clausal form; Is it possible to deduce that 'The book supports the cup'.

Program:

```
above(cup,book). touch(cup,book).

support(Y,X) :-

above(X,Y),touch(X,Y).
```

Output:



Experiment:23

Aim: Solve the classical Water Jug problem of AI.

Program:

```
start(2,0):write(' 4lit Jug: 2 | 3lit Jug:
    0|'),nl, write('~~~~~'),nl,
    write('Goal Reached!
    Congrats!!'),nl, write('~~~~~').
    start(X,Y):write(' 4lit Jug: '),
    write(X),
      write('| 3lit Jug: '),
      write(Y),
      write('|'),nl, write(' Enter the
    move::'), read(N), contains(X,Y,N).
    contains(_,Y,1):-start(4,Y).
    contains(X,_,2):-start(X,3).
    contains(_,Y,3):-start(0,Y).
    contains(X,_,4):-start(X,0).
    contains(X,Y,5):-N is Y-
    4+X,start(4,N). contains(X,Y,6):-N
    is X-3+Y,start(N,3).
    contains(X,Y,7):-N is
    X+Y,start(N,0). contains(X,Y,8):-N
    is X+Y,start(0,N). main():-write('
    Water Jug Game '),nl, write('Intial
```

```
State: 4lit Jug- 0lit'),nl, write(' 3lit
Jug- 0lit'),nl, write('Final
State: 4lit Jug- 2lit'),nl, write(' 3lit
Jug- 0lit'),nl, write('Follow the Rules:
'),nl, write('Rule 1: Fill 4lit Jug'),nl,
write('Rule 2: Fill 3lit Jug'),nl,
write('Rule 3: Empty 4lit Jug'),nl,
write('Rule 4: Empty 3litJug'),nl,
write('Rule 5: Pour water from 3lit
Jug to fill 4lit Jug'),nl, write('Rule 6:
Pour water from 4lit Jug to fill 3lit
Jug'),nl, write('Rule 7: Pour all of
water from 3lit Jug to 4lit Jug'),nl,
write('Rule 8: Pour all of water from
4lit Jug to 3lit Jug'),nl, write(' 4lit Jug:
0 | 3lit Jug: 0'),nl, write(' Enter the
move::'), read(N), contains(0,0,N)
```

Output:

```
main.

Water Jug Garry'
Intial State: 4lit Jug — 0lit
3lit Jug — 0lit
Final State: 4lit Jug — 2lit
3lit Jug — 0lit
Follow the Rules:
Rule 1: Fill 4lit Jug
Rule 2: Fill 3lit Jug
Rule 3: Empty 4lit Jug
Rule 4: Empty 3lit Jug
Rule 5: Pour water from 3lit Jug to lill 4lit Jug
Rule 6: Pour water from 4lit Jug to lill 3lit Jug
Rule 7: Pour all of water from 3lit Jug to 4lit Jug
Rule 8: Pour all of water from 4lit Jug to 3lit Jug
4lit Jug: 0 | 3lit Jug: O
Enter the move::

4lit Jug: 4| 3lit Jug: 0|
Enter the more:;

4lit Jug: 1| 3lit Jug: 3|
Enter the more:;

4lit Jug: 1| 3lit Jug: 0|
Enter the more:;
8
4lit Jug: 0| 3lit Jug: 1|
Enter the more:;

4lit Jug: 4| 3lit Jug: 1|
Enter the more:;

4lit Jug: 2| 3lit Jug: 3|
Enter the more:;

4lit Jug: 2 | 3lit Jug: 0|

Goal Reached! CongraB!!

true
```

Experiment:24

Aim :Solve the classical Monkey Banana problem of AI.

Program:

```
move(state(middle,onbox,middle,hasnot), grasp,
state(middle,onbox,middle,has)). move(state(P,onfloor,P,H), climb,
state(P,onbox,P,H)). move(state(P1,onfloor,P1,H), drag(P1,P2),
state(P2,onfloor,P2,H)). move(state(P1,onfloor,B,H), walk(P1,P2),
state(P2,onfloor,B,H)). canget(state(_,_,_,has)). canget(State1)
:move(State1,_,State2), canget(State2).
```

Output: