SRM
UNIVERSITY
DELHI-NCR, SONEPAT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Practical File

Program	**B.Tech**
Year & Semester	**III & VI**
Name of Student	**Kanishka Gupta**
Registration No.	**10319210040**
Course Code	
Course Title	Artificial intelligence and expert system

CERTIFICATE

Registration Number : 10319210040

Certified that this is a Bonafide record of work done by **Kanishka Gupta** student of course Bachelor of Technology branch Computer Science & Engineering / III Year / VI semester in CS…..- Subject Name during the year 2021-22.

Subject In-charge

(Name of the Teacher)

Submitted to university External practical Examination held on _____conducted by Department of Computer Science & Engineering as per University Examination guidelines.

Internal Examiner External Examiner

I N D E X

Exp no.	Exp. name	Date	Page no.	Sign.
1	Intro to prolog			
2	Steps to install SWI			
3	Implement any 5 prologrules			
4	To check whether element is in list or not			
5	To find length of list			
6	To perform concatenation			
7	To check whether list issorted			
8	To solve tower of hanoi			
9	Implement backtracking			
10	Cut and fail predicate			

AIM: To solve tower of hanoi.

Theory:

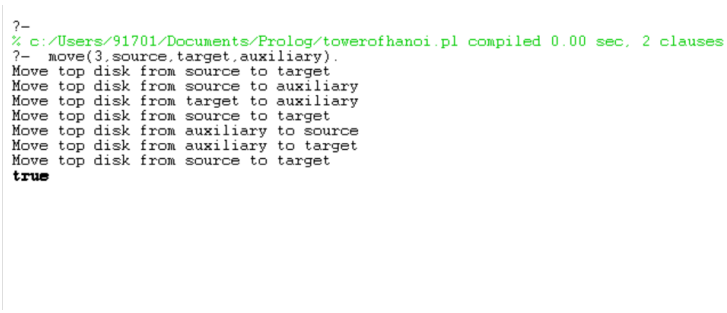
Towers of Hanoi Problem is a famous puzzle to move N disks from the source peg/tower to the target peg/tower using the intermediate peg as an auxiliary holding peg. There are two conditions that are to be followed while solving this problem −

- A larger disk cannot be placed on a smaller disk.
- Only one disk can be moved at a time.

PROGRAM:

```
ove(1,X,Y,_) :-
  write('Move top disk from '), write(X), write(' to '), write(Y), nl.
move(N,X,Y,Z) :-
  N>1, M is N-1,
  move(M,X,Z,Y),
  move(1,X,Y,_), move(M,Z,Y,X).
```

OUTPUT:



AIM: To implement backtracking in prolog.

PROGRAM:

```
boy(tom).
boy(bob).
girl(alice).
girl(lili).
pay(X,Y) :- boy(X), girl(Y)
```

OUTPUT:

```
?-
% c:/Users/91701/Documents/Prolog/lab3.pl compiled 0.00 sec, 1 clauses
?- pay(X,Y).
X = tom,
Y = alice ;

?-
```

AIM: To implement cut and fail predicate in prolog.

Theory:

If we want to restrict backtracking we can control which sub-goals can be redone using the cut !. It succeeds when called, but fails the parent goal (the goal that matched the head of the clause containing the cut) when an attempt is made to redo it on backtracking. It commits to the choices made so far in the predicate. unlimited backtracking can occur before and after the cut but no backtracking can go through it.

PROGRAM:

```
bird(parrot).

bird(penguins).

bird(dove).

bird(robin).

can_fly(penguins) :- !, fail.

can_fly(A) :- bird(A).
```

OUTPUT: