INDEX

Experiment No. 01 - (Section A)

Name of the Experiment

Write a program using PROLOG for the addition and multiplication of two numbers.

Objective

To develop a PROLOG program that accepts two numbers from the user and displays their sum and product.

Software or Tools: SWI-Prolog (or any Prolog interpreter)

Theory

PROLOG is a logic programming language used in the field of artificial intelligence. Unlike procedural languages, PROLOG operates through pattern matching, recursion, and logical inference.

Code

```
% Addition of two numbers----

go:- write('Enter the first number: '),read(X),nl,
   write('Enter the second number: '),read(Y),nl,
   summultiplication(X,Y).

summultiplication(X,Y):-
   S is X+Y,
   M is X*Y,
   write('Summation is :  '),write(S),nl,
   write('Product is :  '),write(M),nl.
```

Output:

```
xiamsec@DARK /media/xiamsec/New Volume/Prolog Lab $ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['AM.pl'].
true.

?- go.
Enter the first number: 4.

Enter the second number: |: 5.

Summation is :  9
Product is :  20
true.

?-
```

Experiment No. 02 - (Section A)

Name of the Experiment

Write a program using PROLOG to determine the Greatest Common Divisor of two positive integer numbers.

Objective

To develop a program using PROLOG/LISP that takes two positive integer numbers as input and computes their Greatest Common Divisor (GCD) using a suitable algorithm such as Euclid's method.

Software or Tools: SWI-Prolog (or any Prolog interpreter)

Theory

PROLOG is a logic programming language used in the field of artificial intelligence. Unlike procedural languages, PROLOG operates through pattern matching, recursion, and logical inference.

Code:

```
go :-
  write("Enter the first input X1: "),
  read(X1),
  write("Enter the second input X2: "),
  read(X2),
  gcd(X1, X2, Result),
  write("GCD is: "), write(Result), nl.

% Base cases
gcd(X, 0, X) :- !.
```

```
gcd(0, X, X) :- !.

% Recursive case using Euclid's algorithm
gcd(X, Y, Result) :-
  X > Y,
  R is X mod Y,
  gcd(Y, R, Result).
gcd(X, Y, Result) :-
  X =< Y,
  gcd(Y, X, Result).
```

Output:

```
xiamsec@DARK ⊏ /media/xiamsec/New Volume/Prolog Lab  swipl Σ ✔ Σ 291 Σ 20:50:40
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['GCD-R.pl'].
true.

?- go.
Enter the first input X1: 36.
Enter the second input X2: |: 60.
GCD is: 12
true .

?-
```

Experiment No:01 - (Section-B)

Name of the Experiment

Write a program using PROLOG to solve the Tower of Hanoi problem.

Objective

To implement a PROLOG program that solves the Tower of Hanoi problem by using a recursive approach to determine the correct sequence of moves.

Software or Tools: SWI-Prolog (or any Prolog interpreter)

Theory

PROLOG is a logic programming language used in the field of artificial intelligence. Unlike procedural languages, PROLOG operates through pattern matching, recursion, and logical inference.

Code:

```
% Base case: Only one disk to move
move(1, X, Y, _) :-
  write('Move top disk from '), write(X), write(' to '), write(Y), nl.

% Recursive case: Move N disks using auxiliary peg
move(N, X, Y, Z) :-
  N > 1,
  M is N - 1,
  move(M, X, Z, Y),     % Step 1: Move top N-1 disks from X to Z using Y
  move(1, X, Y, _),     % Step 2: Move the largest disk from X to Y
  move(M, Z, Y, X).     % Step 3: Move N-1 disks from Z to Y using X
```

Output:

```
xiamsec@DARK E /media/xiamsec/New Volume/Prolog Lab   swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['ToH.pl'].
true.

?- move(3, left, right, center).
Move top disk from left to right
Move top disk from left to center
Move top disk from right to center
Move top disk from left to right
Move top disk from center to left
Move top disk from center to right
Move top disk from left to right
true
```

Experiment No. 02 - (Section B)

Name of the Experiment

Write a program using PROLOG or LISP to solve the Traveling Salesman problem.

Objective

To implement a PROLOG/LISP program that solves the Traveling Salesman Problem (TSP) by finding the shortest possible route that visits each city exactly once and returns to the starting city.

Software or Tools: SWI-Prolog (or any Prolog interpreter)

Theory

PROLOG is a logic programming language used in the field of artificial intelligence. Unlike procedural languages, PROLOG operates through pattern matching, recursion, and logical inference.

Code:

```
road("tampa", "houston", 200).
road("gordon", "tampa", 300).
road("houston", "gordon", 100).
road("houston", "kansas_city", 120).
road("gordon", "kansas_city", 130).

% Direct connection
route(Town1, Town2, Distance) :-
  road(Town1, Town2, Distance).

% Indirect connection with recursion
route(Town1, Town2, Distance) :-
```

```
road(Town1, X, Dist1),
route(X, Town2, Dist2),
Distance is Dist1 + Dist2.
```

Output:

```
xiamsec@DARK /media/xiamsec/New Volume/Prolog Lab  swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['TSP.pl'].
true.

?- route("tampa", "houston", Distance).
Distance = 200 .

?- route("tampa", "kansas_city", Distance).
Distance = 320 .

?- route("gordon", "houston", Distance).
Distance = 500
```