Experiment - 26

Write a prolog Program to sum the Integers from 1 to n.

AIM: To sum integers from 1 to n.

ALGORITHM:

1. Sum (N, Result) :-
2. N > 0, N is +ve.
3. N1 is N-1, % decrement N by 1
4. Sum (N1, SubResult), % Recursive call to sum integers from 1 to N1
5. Result is N + SubResult.

PROGRAM:

```
Sum (0,0). % base case : sum of 0 is 0
Sum(N, Sum) :- N>0,
        N1 is N-1,
        Sum (N1, Sum1),
        Sum is Sum1 + N.
? - Sum (5, Sum).
Sum = 15.
? - Sum (10, Sum).
```

OUTPUT:

Sum = 55

RESULT:

Thus, the program was executed Successfully.

Experiment - 27

Write a Prolog Program for A DB with Name, DOB.

AIM:

To A DB WITH NAME, DOB using Python.

PROGRAM:

```
dob (john, date (1990,5,1)).
dob (jane, date (1985,2,10)).
dob (bob, date (1978,3,28)).
dob (sue, date (1995,8,15)).
dob (tom, date (2000,4,22)).
lookup (Name, DOB) :-
    dob (Name, DOB).
example query : lookup john DOB.
? - lookup (john, DOB).
DOB = date (1990,5,1).
? - lookup (sue, DOB).
DOB = date (1995,8,15).
```

Output:

```
1 ?- cpargs -directory ("D:/TP Prolog/sample code").
Yes
1 ?- [kb1].
Yes
1 ?- girl (priya).
.
Yes
1 ?- girl (vanitha).
no
1 ?- cpr-code (priya).
```

RESULT:

Thus, the program was executed Successfully.

Experiment - 28

Write a Prolog Program for STUDENT-TEACHER-SUB-CODE.

AIM:

To find the Student teacher code

PROGRAM:

```
teacher(john, math).
teacher(jane, english).
teacher(bob, science).
teacher(sue, history).
teacher(tom, art).
taker(alice, math).
taker(alice, science).
taker(bob, english).
taker(bob, science).
taker(carol, history).
taker(carol, art).
taker(dave, math).
taker(dave, english).
taker(dave, art).
teaching_subjects(Teacher, Subject):-
    teacher(Teacher, Subject).
taking_students(Subject, Student):-
    taker(Student, Subject).
?- teaching_subjects(john, Subject).
Subject = math.
?- taking_students(science, Student).
Student = alice;
Student = bob.
% example query: find students taking math
and art
?- taking_students(math, Student), taking_students
(art, Student).
```

Output:

```
Student = alice;
Student = dave;
Student = carol.
```

RESULT:

Hence Prolog Program for STUDENT-TEACHER-SUB-CODE is done.

Experiment : 29

Write a Prolog Program for PLANETS DB

PROGRAM:

```
planet(mercury, rocky, small, hot, closest-to-sun).
planet(venus, rocky, small, hot, 2nd-closest-to-sun).
planet(earth, rocky, medium, temperate, 3rd-closest-sun).
planet(mars, rocky, small, cold, 4th-closest-to-sun).
planet(jupiter, rocky, gas-giant, large, 5th-closest-sun).
planet(saturn, gas-giant, large, cold, 6th-closest-sun).
planet(uranus, ice-giant, large, cold, 7th-closest-sun).
planet(neptune, ice-giant, large, cold, 8th-closest).

planet_properties(name, Type, Size, Temperature, Position):-
planet(name, Type, Size, Temperature, Position).
? - planet_properties(earth, Type, size, temperature).
Type = rocky,
Size = medium,
Temperature = temperate,
Position = 3rd-closest-sun. % org query: find gas giant
? - planet_properties(name, gas-giant, -, -, -).
name = jupiter;
Name = saturn.
? - planet_properties(Name, -, small, hot, -).
```

Output:

```
Name = mercury;
name = venus;
Name = mars.
```

RESULT:

Thus, the program for PLANETS DB is done successfully.

Experiment - 30

Write a Prolog Program to implement Towers of Hanoi.

PROGRAM:

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

OUTPUT:

```
? - hanoi(3, 'A', 'B', 'C').
```

This will solve Towers of Hanoi for 3 disk.

source peg → A
auxiliary peg → B
target peg → C.

RESULT:

Thus, the above program was executed successfully.

Experiment -3)

Write a Prolog Program to print particular bird can fly or not. Incorporate required query.

PROGRAM:

```
bird(ostrich, cannot-fly).
bird(penguin, cannot-fly).
bird(kiwi, cannot-fly).
bird(eagle, can-fly).
bird(pigeon, can-fly).
bird(sparrow, can-fly).
can-fly(Bird):-
    bird(Bird, can-fly).
?- can-fly(eagle)
true.
?- can-fly(ostrich).
false.
?- bird(Bird, cannot-fly).
Bird = ostrich;
Bird = penguin;
Bird = kiwi.
?- bird(Bird, can-fly).
Bird = eagle;
Bird = pigeon;
Bird = sparrow.
```

OUTPUT:

```
yes
| ?- parent(X, jim).
X = pat ? ;
X = peter

yes
| ?-
mother(X, Y).
X = pam
Y = bob ? ;
X = pat
Y = jim ? ;
no
| ?- parent(X).
X = pam ? ;
X = tom ? ;
```

RESULT:

Thus, the Prolog Program was executed successfully.

Experiment : 32

Write the Prolog Program to implement family tree Pam, Liz, Ann and Pat are female, while Tom, Bob and Jim are male persons, Using this information, define the following relations: • Define "mother" relation • Define the "father" relation : • Define "grandfather" relation : • Define "grandmother" relation : • Define "sister" relation • Define "brother" relation.

PROGRAM :

```
female (pam).
female (liz).
female (ann).
female (pat).

male (tom).
male (bob).
male (jim).

parent (tom, liz).
parent (tom, bob).
parent (pam, liz).
parent (pam, bob).
parent (bob, ann).
parent (bob, pat).
parent (liz, jim).
```

```
mother (X,Y) :-
    female (X),
    parent (X,Y).

father (X,Y):-
    male (X),
    parent (X,Y).

grandmother (X,Y):-
    female (X),
    parent (X,Z),
    parent (Z,Y).

grandfather (X,Y):-
    male (X),
    parent (X,Z),
    parent (Z,Y).

sister (X,Y):-
    female (X),
    parent (Z,X),
    parent (Z,Y),
    X\=Y.

Brother (X,Y):-
    male (X),
    parent (Z,X),
    parent (Z,Y),
    X\=Y.
```

Output:

```
1 1  Call : caregt (State (atdoor, onfloor, atwindow, hasnot)) ?
2 2  Call : move (State (atdoor, onfloor, atwindow, _523, _512)) ?
                    ;
                    :
7 4  Exit : cage (State (middle, onbox, middle, hasnot)) ?
```

RESULT:

Thus, the prolog program was executed successfully.

Experiment-33

Write a Prolog program to suggest Diet System based on Disease.

PROGRAM:

```
food(apple, fruit, sweet, low-calorie).
food(banana, fruit, sweet, high-calorie).
food(carrot, vegetable, savory, low-calorie).
food(potato, vegetable, savory, high-calorie).
food(chicken, meat, savory, high-protein).
food(fish, seafood, savory, high-protein).
food(spinach, vegetable, savory, high-iron).
food(almonds, nut, savory, high-fat).

diet(heart-disease, [apple, carrot, chicken, fish, almonds]).
diet(diabetes, [apple, carrot, fish, spinach, almonds]).
diet(anemia, [spinach, chicken, fish, almonds]).
diet(obesity, [apple, carrot, fish, spinach]).

suggest-diet(Disease, Diet):-
    diet(Disease, Diet).
suggest-diet(Disease, Diet):-
    diet(Disease, AllowedFoods),
    findall(Food, (food(Food, _, _, _), member
(Food, AllowedFoods)), Diet).

%.
% Query: suggest-diet(heart-disease, Diet).
% Expected output: Diet = [apple, carrot, chicken, Diet].
%
% Query: suggest-diet(anemia, Diet).
% Expected output: Diet = [spinach, chicken, fish, almonds]
%.
% Query: suggest-diet(obesity, Diet).
% Expected Output: Diet = [apple, carrot, fish, spinach].
```

OUTPUT:

```
?- suggest-diet(hypertension, DietSuggestion).
DietSuggestion = "DASH (Dietary Approaches to Stop
Hypertension) diet. Focus on fruits, vegetables,
whole grains, lean proteins, and low-fat dairy
products. Limit sodium, saturated fat, and added
sugar."
```

RESULT:

Thus, the prolog program was executed successfully.

Experiment : 34

Write a Prolog program to implement Monkey Banana Problem.

PROGRAM:

```
% Define initial state and final state
initial_state (state (at_door, on_floor, at_window,
has_not_eaten)).
final_state (state (_, _, _, has_eaten)).
action (state (at_door, on_floor, at_window, has_not_
eaten), climb, state (at_window, on_floor, has_eaten)
action ( state (at_window, on_window, at_window,
has_not_eaten), climb, state (at_door, on_window, at_door,
has_not_eaten)).
action (state (at_middle, on_floor, at_middle, has_eaten),
walk, state (at_door, on_floor, at_door, has_eaten))
execute_actions (_, [], final state) :- final_state (FinalState
execute_actions (current State, [Action | Rest], FinalState) :-
    action (currentState, Action, NextState),
    execute_actions (NextState, Rest, Final State).
solve_problem (Action List) :-
    initial_state (Initial State),
    execute_actions (Initial State, Action List, FinalSt),
    final_state (Final State).
%
% query : solve_problem (Action List).
% Expected output : ActionList = [climb, grasp, climb, walk, grasp]
```

OUTPUT:

```
yes
{trace}
1 ?- f(1, Y), 2<Y.
  1 1 call : f(1, _23) ?
  2 2 call : 1<3 ?
  2 2 Exit : 1<3 ?
  1 1 Exit : f(1,0) ?
  3 1 call : 2<0 ?
      ;
  1 1 Fail : f(1, _23) ?
(46 ms) no
{trace}
1 ?-
```

RESULT:

Thus, the prolog program was executed successfully.

Experiment - 35

Write a Prolog Program for fruit and its color using Back Tracking.

PROGRAM:

```
% Define possible fruits and colors.
fruit(apple, red).
fruit(banana, yellow).
fruit(grape, purple).
fruit(orange, orange).

match(Fruit, color).
fruits_with_color(Fruit wit, color) :-
    forall(Fruit, match_fruit_color(fruit, color),
           Fruit wit).
% sample queries and expected outputs.
% Query: match_fruit_color(apple, color)
% Expected output: color = red.
%
% Query: match_fruit_color(pear, color).
% Expected output: false.
```

Output:

```
Fruit = apple,
color = red

Fruit = pear,
color = green;
false.
```

RESULT:

Thus, the Prolog Program was executed successfully.

Experiment - 36

Write a Prolog program to implement Best First Search algorithm.

PROGRAM:

```
from queue import priority queue.
V = 14
graph = [[] for i in range (v)]
def best_f_s (actual_src, target, n):
    visited = [false]*n
    pq = priority queue()
    pq.put ((0, actual_src))
    while pq.empty() == False:
    u = pq.get()[1]
    if u == target:
        break.
    for v, c in graph[u]:
        if visited[v] == False:
            visited[v] = True.
            pq.put ((c,v))
        print()
def addedge (x, y, cost):
    graph[x].append ((Y, cost))
    graph[Y].append ((X, cost))

addedge (0, 1, 3)
addedge (0, 2, 6)
addedge (0, 3, 5)
addedge (1, 4, 9)
addedge (1, 5, 8)
addedge (2, 6, 12)
```

```
addedge (3,8,7)
addedge (8,9,5)
addedge (8,10,6)
addedge (9,11,1)
addedge (9,12,10)
addedge (9,13,2)
source = 0
target = 9
Best_first_search (source, target, V)
```

Output:

```
0 1 3 2 8 9
```

RESULT:

Thus, the Prolog program was executed successfully.

Experiment - 37

Write the Prolog Program for Medical Diagnosis.

PROGRAM:

```
implement main
    open core, string
clauses
    run():-
        console::init(),
        succeed().
domains
    disease, indication = symbol
    patient, name = string
predicates
    hypothesis(string, disease)
    symptom(name, indication)
    response(char)
    go
clauses
    go:-
        write("What is patient name?"),
        readln(Patient),
        hypothesis(Patient, Disease),
        write(Patient, " probably has", Disease, "."), nl.

    go:-
        write("Sorry, I don't seem to be able to"), nl,
        write("diagnose disease."), nl.
symptom(Patient, fever):-
        write("Does ", Patient, " have a fever (y/n)?"),
        response(Reply),
        Reply = 'y'.
```

```
symptom(patient, headache) :-
    write("Does", patient, " have a headache (y/n) ?"),
    response(Reply),
    Reply = 'y'.
symptom(patient, runny_nose) :-
    write("Does", patient, " have a conjunctivities (y/n)"),
    response(Reply),
    Reply = 'y'.
symptom(patient, swollen_glands) :-
    write("Does", patient, " have a swollen_glands (y/n ?)"),
    response(Reply),
    Reply = 'y'.
hypothesis(patient, measles) :-
        symptom(patient, fever),
        symptom(patient, cough),
        symptom(patient, conjunctivitis),
        symptom(patient, runny_nose),
        symptom(patient, rash).
hypothesis(patient, common_cold) :-
        symptom(patient, headache),
        symptom(patient, sneezing),
        symptom(patient, chills),
        symptom(patient, runny_nose),
```

```
hypothesis(patient, chicken_pox) :-
        symptom(patient, fever),
        symptom(patient, chills),
        symptom(patient, body_ache),
        symptom(patient, rash).
hypothesis(patient, measles) :-
        symptom(patient, cough),
        symptom(patient, sneezing),
        symptom(patient, runny_nose).
response(Reply) :-
        readchar(Reply),
        write(Reply), nl.
end implement main
goal
    mainExe::run(main::run).
```

Output:

?- best_first_search(a, f, Path).

RESULT:

Thus, the Prolog program for medical Diagnosis was executed successfully.

Experiment -38

Write a Prolog Program for forward chaining. Incorporate required queries.

PROGRAM:

```
bird(penguin) :- cannot_fly.
bird(eagle) :- can_fly.
cannot_fly.
can_fly :- has_wings.
has_wings.
super(X) :- bird(X).
% Example queries
?- super(penguin).
?- super(eagle).
?- super(crow).
```

Result: Hence it is proved.

RESULT:

Hence back tracking verified.

Experiment -39

Write a Prolog Program for backward chaining. Incorporate required queries.

PROGRAM:

```
bird(penguin) :- cannot_fly.
bird(eagle) :- can_fly.
cannot_fly.
can_fly :- has_wings.
has_wings.
super(X) :- bird(X).
super(X) :- can_fly, X = eagle.
super(X) :- cannot_fly, X = penguin.
?- super(crow).
?- super(X).
```

OUTPUT:

True.

RESULT:

Hence back tracking verified.

Experiment -40

Create a Web Blog using word Press to demonstrate Anchor Tag, Title Tag, etc...

AIM:

To use Web Blog Blog using Python Programming

PROGRAM:

```
<UI>
<li> <a href = "#manually-create-anchor-links-
wordpress"> How to manually create Anchor links
in wordpress </a> </li>
<li> <a href = "#anchor-links-wordpress-plugin">
How to create Anchor links in wordpress with a
plugin </a> </li>
links in wordpress with Gutenberg </a> </li>
</ul>
<UI>
<li> <a href = "#manually-create-anchor-links-
wordpress" > How to manually create Anchor links
in WordPress </a> </li>
<li> <a href = "#anchor-links-wordpress-plugin
How to create Anchor links in WordPress with a
plugin </a> </li>
<li> <a href = "#anchor-links-wordpress-gutenberg
How to create Anchor links in wordpress with
Gutenberg </a > </li>.
</ul>
```

```
<title> My Blog page Title </title>
```

⇒ Publish the page and test anchor tag links to make sure they work correctly.

⇒ Add more pages to your word.Press site and use anchor tags and title tags as needed.

RESULT:

Hence Web Blog using word Press to demonstrate Anchor Tag, Title Tag, etc. Web Blog using word press to demonstrate Anchor Tag, Title Tag, etc.. is done successfully.