College code: 6F
PALLAVI ENGINEERING COLLEGE
(UGC AUTONOMOUS)
Accredited by NBA and NAAC with 'A' grade, Approved by AICTE, New Delhi & Affiliated to JNTUH-Hyderabad
Certified by ISO 9001 : 2015 | ISO 14001 : 2015 | ISO 50001 : 2018
Kuntloor(V), Adbullapurmet(M), Near Hayathanagar, R.R. Dist. Hyd - 501505, (T.S.) India

Department of Computer Science and Engineering (AI&ML)

PROLOG
LAB MANUAL

Regulation
PR24

Class: II B.Tech II Semester

Prepared by

G. JYOTHI RANI

Assistant Professor

LAB FACULTY **HOD** **PRINCIPAL**

VISION OF THE INSTITUTE

- To emerge as a global leader in imparting quality technical education emphasizing ethical values for the betterment of the society.

MISSION OF THE INSTITUTE

- To create an excellent teaching learning environment and inculcate the aptitude for research.
- To establish centers of excellence through collaborative initiatives.
- To empower the student community by developing creativity and innovation.

Proposed Vision and Mission of the Department

VISION OF THE DEPARTMENT

- To become a leading centre of excellence in Artificial Intelligence and Machine Learning by fostering innovation, research, and collaboration in diverse areas of computer science. We aim to address global challenges and emerging societal needs through advanced education, cutting-edge technologies, and impactful solutions in AI and ML.

MISSION OF THE DEPARTMENT

- To equip students with the knowledge and skills to solve complex, real-world problems in multidisciplinary fields using AI and ML technologies.
- To foster strong domain expertise and research capabilities, enabling students to pursue challenging careers and advanced education in AI and ML.
- To provide students with a strong sense of ethics, professionalism, and a desire for lifelong learning, enabling them to make significant contributions to both the field and society.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Computer Science and Engineering – Data Science graduate will:

PEO	Statements
PEO1	Graduates will be prepared for a successful career in Computer Science discipline and related industry to meet the needs of the nation and leading industries and also to excel in postgraduate programs.
PEO2	Graduates will continue to learn and apply the acquired knowledge to solve Engineering problems and appreciation of the arts, humanities and social sciences.
PEO3	Graduates will have good and broad scientific and engineering knowledgebase so as to comprehend, analyze, design and create novel products and solutions for real-time applications.
PEO4	Graduates will understand professional and ethical responsibility, develop leadership, utilize membership opportunities, and develop effective communication skills, teamwork skills, multidisciplinary approach and life-long learning required for a successful professional career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

The Computer Science and Engineering – Data Science graduate will be able to:

PSOs	Statements
PSO1	Expertise in different aspects and appropriate models of Data Science and use large data sets to cater for the growing demand for data scientists and engineers in industry.
PSO2	Apply the principles and techniques of database design, administration, and implementation to enhance data collection capabilities and decision-support systems.

Program outcomes:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design / Development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest.

DEPARTMENT OF CSE (AI&ML)

Sub. Code : PAM409PC **Year / Sem: II-II**

Sub. Name : PROLOG LAB **Batch: 2023-2027**

Course Objectives:

The learning objectives of this course are to

S.No	Objectives
1	PROLOG stands for Programming, In Logic — an idea that emerged in the early 1970's to use logic as programming language. The early developers of this idea included Robert Kowaiski at Edinburgh (on the theoretical side), Marrten van Emden at Edinburgh (experimental demonstration) and Alian Colmerauer at Marseilles (implementation). David D.H. Warren's efficient implementation at Edinburgh in the mid -1970's greatly contributed to the popularity of PROLOG. PROLOG is a programming language centered around a small set of basic mechanisms, Including pattern matching, tree based data structuring and automatic backtracking. This Small set constitutes a surprisingly powerful and flexible programming framework. PROLOG is especially well suited for problems that involve objects- in particular, structured objects- and relations between them.

PAM409PC: PROLOG/ LISP/ PYSWIP

B.Tech. II Year II Sem.

L	T	P	C
0	0	2	1

List of Programs:

1. Write simple fact for following:
   A. Ram likes mango.
   B. Seema is a girl.
   C. Bill likes Cindy.
   D. Rose is red.
   E. John owns gold

2. Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

3. Write a program to solve the Monkey Banana problem

4. WAP in turbo prolog for medical diagnosis and show the advantages and disadvantages of green and red cuts.

5. Write a program to solve the 4-Queen problem.

6. Write a program to solve traveling salesman problems.

7. Write a program to solve water jug problems using Prolog.

8. Write simple Prolog functions such as the following. Take into account lists which are too short. -- remove the Nth item from the list. -- insert as the Nth item.

9. Assume the prolog predicate gt(A, B) is true when A is greater than B. Use this predicate to define the predicate addLeaf(Tree, X, NewTree) which is true if NewTree is the Tree produced by adding the item X in a leaf node. Tree and NewTree are binary search trees. The empty treeis represented by the atom nil.

10. Write a Prolog predicate, countLists(Alist, Ne, Nl), using accumulators, that is true when Nl is the number of items that are listed at the top level of Alist and Ne is the number of empty lists. Suggestion: First try to count the lists, or empty lists, then modify by adding the other counter.

11. Define a predicate memCount(AList,Blist,Count) that is true if Alist occurs Count times within Blist. Define without using an accumulator. Use "not" as defined in utilities.pro, to make similarcases are unique, or else you may get more than one count as an answer.

    Examples:
    memCount(a,[b,a],N). N
    = 1 ;
    no
    memCount(a,[b,[a,a,[a],c],a],N). N
    = 4 ;
    no
    memCount([a],[b,[a,a,[a],c],a],N). N
    = 1 ;
    No

REFERENCE BOOK:

1. PROLOG: Programming for Artificial Intelligence, 3e, by BRATKO, WILEY

1. **Write simple fact for following:**
   A. **Ram likes mango.**
   B. **Seema is a girl.**
   C. **Bill likes Cindy.**
   D. **Rose is red.**
   E. **John owns gold**

Aim:

Write simple fact for following: A. Ram likes mango.B. Seema is a girl.C. Bill likes Cindy.D. Rose is red.E. John owns gold

Solution :

```
% Facts

1. Ram likes mango.
2. Seema is a girl.
3. Bill likes Cindy.
4. Rose is red.
5. John owns gold.

% Clauses

likes(ram ,mango).
girl(seema).
red(rose).
likes(bill ,cindy).
owns(john ,gold).
```

Output:

```
% Queries

?-likes(ram,What).
What= mango
?-likes(Who,cindy).
Who= cindy

?-red(What).
What= rose
```

```
?-owns(Who,What).
Who= john
What= gold
```

2. Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

Aim:

Write predicates one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

Solution :

```
% Production rules:

c_to_f
f is c * 9 / 5 + 32
freezing f < = 32

% Rules:

c_to_f(C,F) :-
        F is C * 9 / 5 + 32.

freezing(F) :-
        F =< 32.
```

Output:

```
% Queries :

?- c_to_f(100,X).
X = 212
Yes
?- freezing(15).
Yes
?- freezing(45).
No
```

3. Write a program to solve the Monkey Banana problem

Aim:

Write a program to solve the Monkey Banana problem

Solution :

```
/* Description:

Imagine a room containing a monkey, chair and some bananas. That have been hanged from the cen
ter of ceiling. If the monkey is clever enough he can reach the bananas by placing the chair directly
below the bananas and climb on the chair .

The problem is to prove the monkey can reach the bananas.

The monkey can perform the following actions:
1) Walk on the floor
2) Climb the box
3) Push the box around(if it is beside the box).
4) Grasp the banana if it is standing on the box directly under the banana.

*/

% Production rules:

can_reach □ clever,close.
get_on: □ can_climb.

under □ in room,in_room, in_room,can_climb.
Close □ get_on,under | tall

% Clauses:

in_room(bananas).
in_room(chair).
in_room(monkey).
clever(monkey).
can_climb(monkey, chair).
tall(chair).
can_move(monkey, chair, bananas).
can_reach(X, Y):-clever(X),close(X, Y).
get_on(X,Y):-
        can_climb(X,Y).
under(Y,Z):-
        in_room(X),in_room(Y),
        in_room(Z),can_climb(X,Y,Z).
close(X,Z):-
        get_on(X,Y), under(Y,Z);
        tall(Y).
```

Output:

```
% Queries:

?- can_reach(A, B).
A = monkey.
B = banana.

?- can_reach(monkey, banana).
Yes.
```

4. WAP in turbo prolog for medical diagnosis and show the advantages and disadvantages of green and red cuts.

Aim:

WAP in turbo prolog for medical diagnosis and show the advantages and disadvantages of green and red cuts.

Solution :

```
/* Description:

This object of this famous puzzle is to move N disks from the left peg to the right peg using the cent
er peg as an auxiliary holding peg. At no time can a larger disk be placed upon a smaller disk. The f
ollowing diagram depicts the starting setup for N=3 disks.
*/

% Production rules:

hanoi(N) □ move(N,left,middle,right).
move(1,A,_,C) □ inform(A,C),fail.
move(N,A,B,C) □ N1=N-1,move(N1,A,C,B),inform(A,C),move(N1,B,A,C).


% Domains:

loc =right;middle;left

% Predicates:

hanoi(integer)
move(integer,loc,loc,loc)
inform(loc,loc)

% Clauses:
```

```
hanoi(N):-
        move(N,left,middle,right).

move(1,A,_,C):-
        inform(A,C),!.

move(N,A,B,C):-
        N1=N-1,
        move(N1,A,C,B),
        inform(A,C),
        move(N1,B,A,C).

inform(Loc1, Loc2):-
        write("\nMove a disk from ", Loc1, " to ", Loc2).
```

Output:

```
% Queries:

 ?- can_reach(A, B).
A = monkey.
B = banana.

 ?- can_reach(monkey, banana).
Yes.
```

5. Write a program to solve 4-Queens problem

Aim:

Write a program to solve 4-Queens problem

Solution :

```
/* Description:

In the 4 Queens problem the object is to place 4 queens on a chessboard in such a way that no queen
s can capture a piece. This means that no two queens may be placed on the same row, column, or di
agonal.

*/

% Domains:

queen = q(integer, integer)
queens = queen*
freelist = integer*
board = board(queens, freelist, freelist, freelist, freelist)

% Predicates:

nondeterm placeN(integer, board, board)
nondeterm place_a_queen(integer, board, board)
nondeterm nqueens(integer)
nondeterm makelist(integer, freelist)
nondeterm findandremove(integer, freelist, freelist)
nextrow(integer, freelist, freelist)

% Clauses

nqueens(N):-
        makelist(N,L),
        Diagonal=N*2-1,
        makelist(Diagonal,LL),
        placeN(N,board([],L,L,LL,LL),Final),
        write(Final).

placeN(_,board(D,[],[],D1,D2),board(D,[],[],D1,D2)):-!.

placeN(N,Board1,Result):-
        place_a_queen(N,Board1,Board2),
        placeN(N,Board2,Result).

place_a_queen(N,
                board(Queens,Rows,Columns,Diag1,Diag2),
                board([q(R,C)|Queens],NewR,NewC,NewD1,NewD2)):-
nextrow(R,Rows,NewR),
findandremove(C,Columns,NewC),
D1=N+C-R,findandremove(D1,Diag1,NewD1),
```

```
D2=R+C-1,findandremove(D2,Diag2,NewD2).

findandremove(X,[X|Rest],Rest).
findandremove(X,[Y|Rest],[Y|Tail]):-
        findandremove(X,Rest,Tail).

makelist(1,[1]).
makelist(N,[N|Rest]) :-
        N1=N-1,makelist(N1,Rest).
nextrow(Row,[Row|Rest],Rest).
```

Output:

```
% Goal

nqueens(4),nl.
board([q(1,2),q(2,4),q(3,1),q(4,3),[],[],[7,4,1],[7,4,1])
yes
```

6. Write a program to solve Traveling salesman problems

Aim:

Write a program to solve Traveling salesman problems

Solution :

```
/* Description:
For example, there are four cities(Kansas City,Houston,Gordon and Tampa).

-> The distance between Kansas City and Houston is 120.
-> The distance between Kansas City and Tampa is 80.
-> The distance between Houston and Gordon is 100.

*/

% Production Rules:-
 route(Town1,Town2,Distance)□ road(Town1,Town2,Distance).
 route(Town1,Town2,Distance)□ road(Town1,X,Dist1),
 route(X,Town2,Dist2),
 Distance=Dist1+Dist2,

% Domains

 town = symbol
 distance = integer

% Predicates

 nondeterm road(town,town,distance)
 nondeterm route(town,town,distance)

% Clauses

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

 route(Town1,Town2,Distance):-
          road(Town1,Town2,Distance).

 route(Town1,Town2,Distance):-
          road(Town1,X,Dist1),
          route(X,Town2,Dist2),
          Distance=Dist1+Dist2,
          !.
```

Output:

```
% Goal
 route("tampa", "kansas_city", X),
 write("Distance from Tampa to Kansas City is ",X),nl.

Distance from Tampa to Kansas City is 320
X=320
1 Solution
```

7. Write a program to solve water jug problems using Prolog

Aim:

Write a program to solve water jug problems using Prolog

Solution :

```
/* Description:
"You are given two jugs, a 4-gallon one and a 3-gallon one. Neither have any measuring markers on
it. There is a tap that can be used to fill the jugs with water. How can you get exactly 2 gallons of w
ater into the 4-gallon jug?".
*/

/* Production Rules:-
R1: (x,y) --> (4,y) if x < 4
R2: (x,y) --> (x,3) if y < 3
R3: (x,y) --> (x-d,y) if x > 0
R4: (x,y) --> (x,y-d) if y > 0
R5: (x,y) --> (0,y) if x > 0
R6: (x,y) --> (x,0) if y > 0
R7: (x,y) --> (4,y-(4-x)) if x+y >= 4 and y > 0
R8: (x,y) --> (x-(3-y),y) if x+y >= 3 and x > 0
R9: (x,y) --> (x+y,0) if x+y =< 4 and y > 0
R10: (x,y) --> (0,x+y) if x+y =< 3 and x > 0
*/

%database
   visited_state(integer,integer).

%predicates
   state(integer,integer).

%clauses
   state(2,0).

state(X,Y):- X < 4,
   not(visited_state(4,Y)),
   assert(visited_state(X,Y)),
   write("Fill the 4-Gallon Jug: (",X,",",Y,") --> (", 4,",",Y,")\n"),
   state(4,Y).

   state(X,Y):- Y < 3,
         not(visited_state(X,3)),
         assert(visited_state(X,Y)),
         write("Fill the 3-Gallon Jug: (", X,",",Y,") --> (", X,",",3,")\n"),
         state(X,3).
```

```
state(X,Y):- X > 0,
        not(visited_state(0,Y)),
        assert(visited_state(X,Y)),
        write("Empty the 4-Gallon jug on ground: (", X,",",Y,") --> (", 0,",",Y,")\n"),
        state(0,Y).

state(X,Y):- Y > 0,
        not(visited_state(X,0)),
        assert(visited_state(X,0)),
        write("Empty the 3-Gallon jug on ground: (", X,",",Y,") --> (", X,",",0,")\n"),
        state(X,0).

state(X,Y):- X + Y >= 4,
        Y > 0,
        NEW_Y = Y - (4 - X),
        not(visited_state(4,NEW_Y)),
        assert(visited_state(X,Y)),
        write("Pour water from 3-Gallon jug to 4-gallon until it is full: (", X,",",Y,") --> (", 4,",",NE
W_Y,")\n"),
        state(4,NEW_Y).


 state(X,Y):- X + Y >=3,
        X > 0,
        NEW_X = X - (3 - Y),
        not(visited_state(X,3)),
        assert(visited_state(X,Y)),
        write("Pour water from 4-Gallon jug to 3-gallon until it is full: (", X,",",Y,") --> (", NEW_X
,",",3,")\n"),
        state(NEW_X,3).

 state(X,Y):- X + Y>=4,
        Y > 0,
        NEW_X = X + Y,
        not(visited_state(NEW_X,0)),
        assert(visited_state(X,Y)),
        write("Pour all the water from 3-Gallon jug to 4-gallon: (", X,",",Y,") --> (", NEW_X,",",0,"
)\n"),
        state(NEW_X,0).

 state(X,Y):- X+Y >=3,
        X > 0,
        NEW_Y = X + Y,
        not(visited_state(0,NEW_Y)),
        assert(visited_state(X,Y)),
        write("Pour all the water from 4-Gallon jug to 3-gallon: (", X,",",Y,") --> (", 0,",",NEW_Y,"
)\n"),
        state(0,NEW_Y).

 state(0,2):- not(visited_state(2,0)),
        assert(visited_state(0,2)),
        write("Pour 2 gallons from 3-Gallon jug to 4-gallon: (", 0,",",2,") --> (", 2,",",0,")\n"),
```

```
        state(2,0).

    state(2,Y):- not(visited_state(0,Y)),
        assert(visited_state(2,Y)),
        write("Empty 2 gallons from 4-Gallon jug on the ground: (", 2,",",Y,") --> (", 0,",",Y,")\n"),
        state(0,Y).

goal:-
            makewindow(1,2,3,"4-3 Water Jug Problem",0,0,25,80),
            state(0,0).
```

Output:

```
% Goal:-
            makewindow(1,2,3,"4-3 Water Jug Problem",0,0,25,80),
            state(0,0).
+----------------------------4-3 Water Jug Problem-------------------------+
| Fill the 4-Gallon Jug: (0,0) --> (4,0)                                    |
| Fill the 3-Gallon Jug: (4,0) --> (4,3)                                    |
| Empty the 4-Gallon jug on ground: (4,3) --> (0,3)                         |
| Pour all the water from 3-Gallon jug to 4-gallon: (0,3) --> (3,0)         |
| Fill the 3-Gallon Jug: (3,0) --> (3,3)                                    |
| Pour water from 3-Gallon jug to 4-gallon until it is full: (3,3) --> (4,2)  |
| Empty the 4-Gallon jug on ground: (4,2) --> (0,2)                         |
| Pour all the water from 3-Gallon jug to 4-gallon: (0,2) --> (2,0)         |
|                                                                           |
| Press the SPACE bar
```

8. Write simple Prolog functions such as the following. Take into account lists which are too short.-- remove the Nth item from the list. -- insert as the Nth item.

Aim:

Write simple Prolog functions such as the following. Take into account lists which are too short.-- remove the Nth item from the list. -- insert as the Nth item.

Solution:

```
<br />
<b>Warning</b>:  include(ai/week9.pl): failed to open stream: No such file or directory in <b>/ho
me/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line <b>
113</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/week9.pl' for inclusion (include_path='.:/opt/alt/php
73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprograms/sdc
display.php</b> on line <b>113</b><br />
```

Output:

```
<br />
<b>Warning</b>:  include(ai/outputs/w9.txt): failed to open stream: No such file or directory in <b
>/home/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line
<b>165</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/outputs/w9.txt' for inclusion (include_path='.:/opt/a
lt/php73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprogra
ms/sdcdisplay.php</b> on line <b>165</b><br />
```

9. Assume the prolog predicate gt(A, B) is true when A is greater than B. Use this predicate todefine the predicate addLeaf(Tree, X, NewTree) which is true if NewTree is the Treeproducedby adding the item X in a leaf node. Tree and NewTree are binary search trees. Theempty treeis represented by the atom nil.

Aim:

Assume the prolog predicate gt(A, B) is true when A is greater than B. Use this predicate todefine the predicate addLeaf(Tree, X, NewTree) which is true if NewTree is the Treeproducedby adding the item X in a leaf node. Tree and NewTree are binary search trees. Theempty treeis represented by the atom nil.

Solution :

```
br />
<b>Warning</b>:  include(ai/week10.pl): failed to open stream: No such file or directory in <b>/ho
me/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line <b>
113</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/week10.pl' for inclusion (include_path='.:/opt/alt/ph
p73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprograms/sd
cdisplay.php</b> on line <b>113</b><br />
```

Output:

```
<br />
<b>Warning</b>:  include(ai/outputs/w10.txt): failed to open stream: No such file or directory in <
b>/home/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line
<b>165</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/outputs/w10.txt' for inclusion (include_path='.:/opt/
alt/php73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprogra
ms/sdcdisplay.php</b> on line <b>165</b><br />
```

10. Write a Prolog predicate, countLists(Alist, Ne, Nl), using accumulators, that is true when Nl isthe number of items that are listed at the top level of Alist and Ne is the number of emptylists. Suggestion: First try to count the lists, or empty lists, then modify by adding the othercounter.

Aim:

Write a Prolog predicate, countLists(Alist, Ne, Nl), using accumulators, that is true when Nl isthe number of items that are listed at the top level of Alist and Ne is the number of emptylists. Suggestion: First try to count the lists, or empty lists, then modify by adding the othercounter.

Solution :

```
br />
<b>Warning</b>:  include(ai/week11.pl): failed to open stream: No such file or directory in <b>/ho
me/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line <b>
113</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/week11.pl' for inclusion (include_path='.:/opt/alt/ph
p73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprograms/sd
cdisplay.php</b> on line <b>113</b><br />
```

Output:

```
<br />
<b>Warning</b>:  include(ai/outputs/w11.txt): failed to open stream: No such file or directory in <
b>/home/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php</b> on line
<b>165</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'ai/outputs/w11.txt' for inclusion (include_path='.:/opt/
alt/php73/usr/share/pear') in <b>/home/u681245571/domains/studyglance.in/public_html/labprogra
```

ms/sdcdisplay.php on line 165

11. Define a predicate memCount(AList,Blist,Count) that is true if Alist occurs Count times within Blist. Define without using an accumulator. Use "not" as defined in utilities.pro, to make similarcases are unique, or else you may get more than one count as an answer.

Examples:
memCount(a,[b,a],
N). N = 1 ;
no
memCount(a,[b,[a,a,[a],c],a
],N). N = 4 ;
no
memCount([a],[b,[a,a,[a],c],
a],N). N = 1 ;
No

Aim:

Define a predicate memCount(AList,Blist,Count) that is true if Alist occurs Count timeswithin Blist. Define without using an accumulator.

Solution :

Warning: include(ai/week12.pl): failed to open stream: No such file or directory in /home/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php on line 113

Warning: include(): Failed opening 'ai/week12.pl' for inclusion (include_path='.:/opt/alt/php73/usr/share/pear') in /home/u681245571/domains/studyglance.in/public_html/labprograms/sdcdisplay.php on line 113
