

Practical-2

2. Write a program to implement “cut” and “fail” predicate in PROLOG.

Pro-1 :- Implement Minimum and Maximum Without Using Cut And Fail.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Error Correction Line 6 Col 16 C:\WORK.PRO
/* 19BECE30094 PR-2*/
predicates
    larger(integer, integer, integer)
clauses
    larger(A,B,A):- A>B.
    larger(A,B,B).

Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal:
  
```

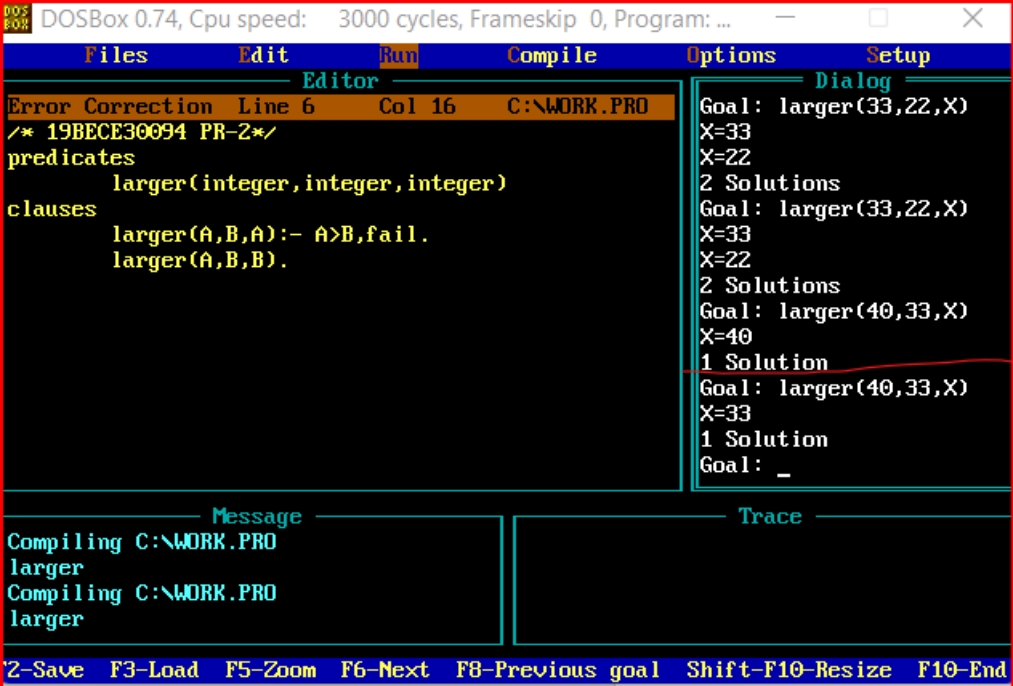
Pro-2 :- Implement Minimum and Maximum With Using Cut And Fail

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Error Correction Line 6 Col 16 C:\WORK.PRO
/* 19BECE30094 PR-2*/
predicates
    larger(integer, integer, integer)
clauses
    larger(A,B,A):- A>B,!.
    larger(A,B,B).

Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal: larger(40,33,X)
X=40
1 Solution
Goal:

Message
Compiling C:\WORK.PRO
Compiling C:\WORK.PRO
Compiling C:\WORK.PRO
larger
Trace
  
```



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...

Files Edit Run Compile Options Setup

Editor

Error Correction Line 6 Col 16 C:\WORK.PRO

```
/* 19BECE30094 PR-2 */
predicates
    larger(integer, integer, integer)
clauses
    larger(A,B,A):- A>B,fail.
    larger(A,B,B).
```

Dialog

Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal: larger(33,22,X)
X=33
X=22
2 Solutions
Goal: larger(40,33,X)
X=40
1 Solution
Goal: larger(40,33,X)
X=33
1 Solution
Goal: _

Message

Compiling C:\WORK.PRO
larger
Compiling C:\WORK.PRO
larger

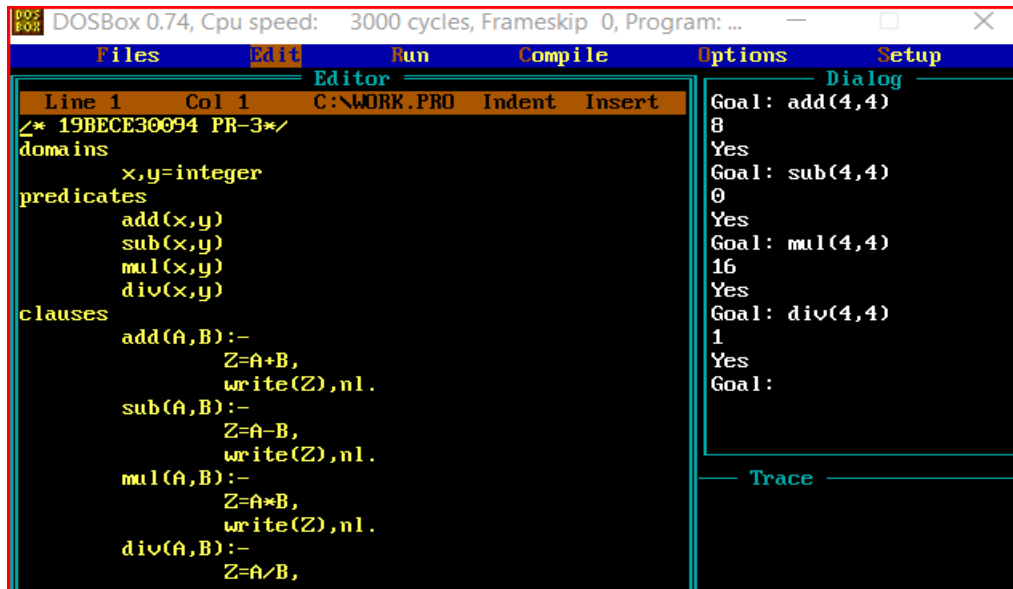
Trace

*2-Save F3-Load F5-Zoom F6-Next F8-Previous goal Shift-F10-Resize F10-End

Practical-3

3. Write a program to implement arithmetic operators, simple input/output and compound goals in PROLOG.

Pro 1:-Implement Arithmetic Operator.



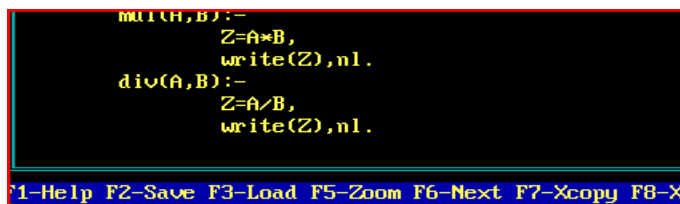
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 1 Col 1 C:\WORK.PRO Indent Insert
/* 19BECE30094 PR-3*/
domains
    x,y=integer
predicates
    add(x,y)
    sub(x,y)
    mul(x,y)
    div(x,y)
clauses
    add(A,B):-
        Z=A+B,
        write(Z),nl.
    sub(A,B):-
        Z=A-B,
        write(Z),nl.
    mul(A,B):-
        Z=A*B,
        write(Z),nl.
    div(A,B):-
        Z=A/B,

```

Goal: add(4,4)
8
Yes
Goal: sub(4,4)
0
Yes
Goal: mul(4,4)
16
Yes
Goal: div(4,4)
1
Yes
Goal:

Trace



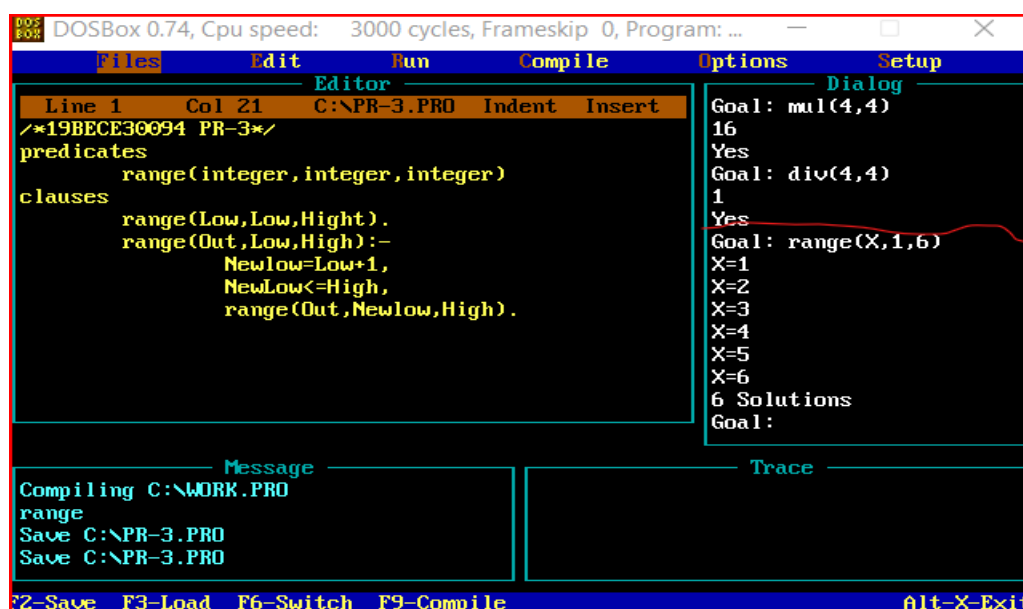
```

mul(A,B):-
    Z=A*B,
    write(Z),nl.
div(A,B):-
    Z=A/B,
    write(Z),nl.

```

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-X

Pro 2:- To print the value between Range.



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 1 Col 21 C:\PR-3.PRO Indent Insert
/*19BECE30094 PR-3*/
predicates
    range(integer, integer, integer)
clauses
    range(Low,Low,High).
    range(Out,Low,High):-
        NewLow=Low+1,
        NewLow<=High,
        range(Out,NewLow,High).

```

Goal: mul(4,4)
16
Yes
Goal: div(4,4)
1
Yes
Goal: range(X,1,6)
X=1
X=2
X=3
X=4
X=5
X=6
6 Solutions
Goal:

Message

Compiling C:\WORK.PRO
range
Save C:\PR-3.PRO
Save C:\PR-3.PRO

Trace

F2-Save F3-Load F6-Switch F9-Compile Alt-X-Exit

Pro 3:-Implement Tower Of hanoi.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 1 Col 1 C:\PR-3-3.PRO Indent Insert
/*19BECE30094 PR-3*/
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):-nl,
        write("Move a disk from", Loc1, "to", Loc2).
inform

```

Goal: hanoi(3)

Move a disk from left to right
 ghtYes
 Goal: hanoi(3)

Move a disk from left to right
 Move a disk from left to middle
 Move a disk from right to middle
 Move a disk from left to right
 Move a disk from middle to left
 Move a disk from middle to right
 Move a disk from left to right
 ghtYes
 Goal:

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-Xedit F9-Compile F10-Menu

Pro 4:-To Find grade.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 12 Col 23 C:\PR-3-4.PRO Indent Insert
/*19BECE30094 PR-3 */
predicates
    grade(integer)
clauses
    grade(M):-
        M<100, M>=70, write("A grade"), nl.
    grade(M):-
        M<70, M>=60, write("B grade"), nl.
    grade(M):-
        M<60, M>=35, write("C grade"), nl.
    grade(M):-
        M<35, write("F"), nl.

```

Save C:\PR-3-4.PRO
 Save C:\PR-3-4.PRO
 Compiling C:\PR-3-4.PRO
 grade

ghtYes
 Goal: hanoi(3)

Move a disk from left to right
 Move a disk from left to middle
 Move a disk from right to middle
 Move a disk from left to right
 Move a disk from middle to left
 Move a disk from middle to right
 Move a disk from left to right
 ghtYes
 Goal: grade(91)
 A grade
 Yes
 Goal: _

F2-Save F3-Load F5-Zoom F6-Next F8-Previous goal Shift-F10-Resize F10-End

Practical-4

4. Write a program to implement recursion in PROLOG.

Pro 1:-Implement Factorial Program.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Line 7 Col 21 C:\AIPR-4(I.PRO Indent Inse
/*19BECE30094 PR-4*/
domains
    n,f=integer
predicates
    factorial(n,f)
clauses
    factorial(0,1).
    factorial(N,F):-
        N1=N-1,
        factorial(N1,F1),
        F=N*F1.

Goal: factorial(6,X)
X=720
1 Solution
Goal: factorial(0,X)
X=1
1 Solution
Goal: factorial(6,X)
X=720

1002 Stack overflow. Re-configure with Options if necessary.
Press the SPACE bar

F1>Error explanation F2-Save F3-Load F5-Zoom F6-Next F10-Continue

```

Pro 2:-Implement Factorial Program using Cut and Fail.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Line 1 Col 3 C:\AIPR-4II.PRO Indent Inse
/*19BECE30094 PR-4*/
domains
    n,f=integer
predicates
    factorial(n,f)
clauses
    factorial(0,1) if !.
    factorial(N,F):-
        N1=N-1,
        factorial(N1,F1),
        F=N*F1.

Goal: factorial(6,X)
X=720
1 Solution
Goal: factorial(0,X)
X=1
1 Solution
Goal:

Load WORK.PRO
Load C:\AIPR-4II.PRO
Compiling C:\AIPR-4II.PRO
factorial

F2-Save F3-Load F5-Zoom F6-Next F8-Previous goal Shift-F10-Resize F10-End

```

Practical-5

5. Write a program to implement Lists in PROLOG.

Program (1): Code for Prolog program to check whether a given list is palindrome or not in Artificial Intelligence

domains

list=symbol*

predicates

palin(list)

findrev(list,list,list)

compare(list, list)

clauses

palin(List1):-

findrev(List1,[],List2), compare(List1, List2).

findrev([],List1, List1).

findrev([X|Tail], List1,List2):-

findrev(Tail,[X|List1], List2).

compare([],[]):-

write("\nList is Palindrome").

compare([X|List1],[X|List2]):-

compare(List1, List2).

compare([X|List1],[Y|List2]):-

write("\nList is not Palindrome").

Output:-

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 1 Col 1 C:\PR-5-1.PRO Indent Insert
/* 19BECE30094 PR-5*/
domains
list = symbol*
predicates
palin(list)
findrev(list,list,list)
compare(list,list)
clauses
palin(List1):-
findrev(List1,[],List2), compare(List1,List2).

findrev([],List1,List2).
findrev([X:Tail],List1,List2):-
findrev(Tail,[X:List1],List2).

compare([],[]):-
write("\n List is plaindrome").

compare([X:List1],[X:List2]):-
compare(List1,List2).
F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-Xedit F9-Compile F10-Menu

```

```

Dialog
Goal: palin([m,i,t]).
List is not PalindromeYes
Goal: palin([n,y,n]).
List is PalindromeYes
Goal:

```

Program (2): Code for Prolog program to reverse a list using concatenate in Artificial Intelligence

domains

x = integer

I = integer*

predicates

reverse(1,1)

concatenate(1,1,1)

clauses

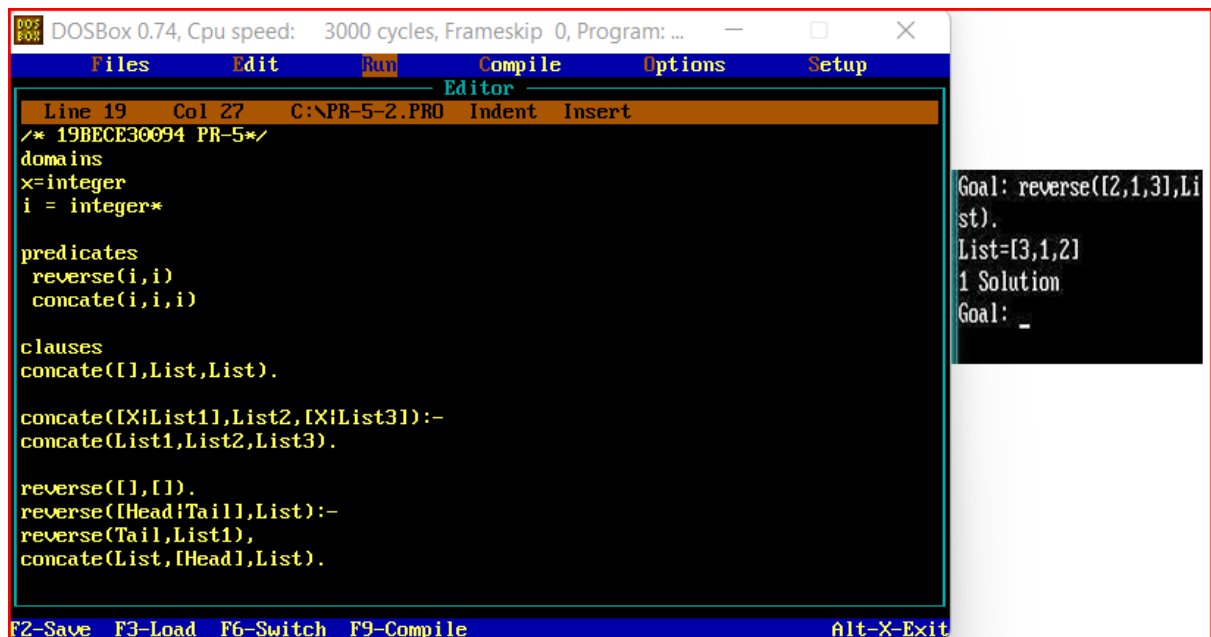
concatenate([],List, List).

concatenate([X|List1], List2, [X|List3]) :-
concatenate(List1, List2, List3).

reverse([],[]).

reverse([Head |Tail], List) :-
reverse(Tail, List1),
concatenate(List1,[Head],List).

Output:-



The screenshot shows a DOSBox window titled "DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...". The window contains a text editor with the following Prolog code:

```
/* 19BECE30094 PR-5*/  
domains  
x=integer  
i = integer*  
  
predicates  
reverse(i,i)  
concat(i,i,i)  
  
clauses  
concat([],List,List).  
  
concat([X|List1],List2,[X|List3]):-  
concat(List1,List2,List3).  
  
reverse([],[]).  
reverse([Head|Tail],List):-  
reverse(Tail,List1),  
concat(List,[Head],List).
```

At the bottom of the editor, the following text is displayed:

```
F2-Save F3-Load F6-Switch F9-Compile Alt-X-Exit
```

To the right of the editor, the output of the program is shown:

```
Goal: reverse([2,1,3],Li  
st).  
List=[3,1,2]  
1 Solution  
Goal: _
```


Practical-6

6. Write a program to implement string operations in PROLOG. Implement string operations like substring, String position, palindrome etc.)

Pro-1. ConcatString

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Error Correction Line 6 Col 19 C:\PR-6-1.PRO
/* 19BECE30094 PR-6*/
predicates
go
clauses
go:-
X="Hello World! ",
Y="Prolog",
concat(X,Y,Z),
write(Z),nl.

Message
Compiling C:\PR-6-1.PRO
Compiling C:\PR-6-1.PRO
Compiling C:\PR-6-1.PRO
go

Dialog
Goal: go
Hello World! Prolog
Yes
Goal:

```

F2-Save F3-Load F5-Zoom F6-Next F8-Previous goal Shift-F10-Resize F10-End

Pro-2. Split String:- frontstr().

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Error Correction Line 5 Col 5 C:\PR-6-2.PRO
/* 19BECE30094 PR-6*/
predicates
go
clauses
go:-
X="Hello World! Prolog",
frontstr(12,X,Select,Rest),
write(Select),nl,
write(Rest),nl.

Message
Compiling C:\PR-6-2.PRO
Compiling C:\PR-6-2.PRO
Compiling C:\PR-6-2.PRO
go

Dialog
Goal: go
Hello World!
Prolog
Yes
Goal:

```

F2-Save F3-Load F5-Zoom F6-Next F8-Previous goal Shift-F10-Resize F10-End

Pro-3. Find Length Of String :- str_len().

The screenshot shows a DOSBox window with a Prolog editor. The program file is C:\PR-6-3.PRO. The code in the editor is as follows:

```

Line 8 Col 13 C:\PR-6-3.PRO Indent Insert
/* 19BECE30094 PR-6*/
predicates
go
clauses
go:-
X="Hello World! Prolog",
str_len(X,Length),
write(Length),nl.

```

The output window on the right shows the following interaction:

```

Goal: go
Hello World!
Prolog
Yes
Goal: go
19
Yes
Goal:

```

The message window at the bottom shows the compilation and execution steps:

```

Save C:\PR-6-3.PRO
Save C:\PR-6-3.PRO
Compiling C:\PR-6-3.PRO
go

```

The status bar at the bottom indicates keyboard shortcuts: F2-Save, F3-Load, F5-Zoom, F6-Next, F8-Previous goal, Shift-F10-Resize, F10-End.

Pro-4. Converting upper_lower()

The screenshot shows a DOSBox window with a Prolog editor. The program file is C:\PR-6-4.PRO. The code in the editor is as follows:

```

Line 7 Col 38 C:\PR-6-4.PRO Indent Insert
/* 19BECE30094 PR-6*/
predicates
go
clauses
go:-
X="ketul shah",
upper_lower(Upper,X),write(Upper),nl,
upper_lower(X,Lower) , write(Lower),nl.

```

The output window on the right shows the following interaction:

```

Goal: go
Hello World!
Prolog
Yes
Goal: go
19
Yes
Goal: go
KETUL SHAH
ketul shah
Yes
Goal:

```

The message window at the bottom shows the compilation and execution steps:

```

Compiling C:\PR-6-4.PRO
Save C:\PR-6-4.PRO
Compiling C:\PR-6-4.PRO
go

```

The status bar at the bottom indicates keyboard shortcuts: F2-Save, F3-Load, F5-Zoom, F6-Next, F8-Previous goal, Shift-F10-Resize, F10-End.

Practical-7

7. Write a prolog program to maintain family tree.

Program : Code for Prolog program for family hierarchy in Artificial Intelligence

predicates

male(symbol).

female(symbol).

father(symbol,symbol).

husband(symbol,symbol).

brother(symbol,symbol).

sister(symbol,symbol).

listbrothers(symbol).

listsisters(symbol).

mother(symbol,symbol).

grandfather(symbol).

grandmother(symbol).

uncle(symbol).

aunt(symbol).

cousin(symbol).

listgrandsons(symbol).

listgranddaughters(symbol).

printmenu.

action(integer).

repeat.

clauses

male(dashrath).

male(ram).

male(laxman).

male(bharat).

male(luv).

male(kush).
male(son_of_laxman).
female(kaushalya).
female(sita).
female(urmila).
female(daughter_of_dashrath).
father(dashrath,ram).
father(dashrath,laxman).
father(dashrath,bharat).
father(ram,luv).
father(ram,kush).
father(laxman,son_of_laxman).
father(dashrath,daughter_of_dashrath).
husband(dashrath,kaushalya).
husband(ram,sita).
husband(laxman,urmila).
mother(X,Y):- husband(Z,X),
father(Z,Y).
brother(X,Y):- father(Z,X),
father(Z,Y),
X<>Y,
male(X).
sister(X,Y):- father(Z,X),
father(Z,Y),
X<>Y,
female(X).
listbrothers(X) :- brother(Z,X),
write(Z).
listsisters(X):- sister(Z,X),
write(Z).

```
grandfather(X):- father(Y, Z),  
father(Z,X),  
write(Y, " is the grandfather of ",X,"\n").  
grandmother(X):- husband(Z,X),  
father(Z,V),  
father(V,Y),  
write(Y, " is the grandmother of ",X,"\n").
```

```
listgrandsons(X):- father(X,Z),  
father(Z,Y),  
male(Y),  
write(Y,"\n"),  
fail.
```

```
listgrandsons(X):- husband(Y,X),  
father(Y,V),  
father(V,Z),  
male(Z),  
write(Z,"\n"),  
fail.
```

```
listgranddaughters(X):- father(X,Z),  
father(Z,Y),  
female(Y),  
write(Y,"\n"),  
fail.
```

```
listgranddaughters(X):- husband(Y,X),  
father(Y,V),  
father(V,Z),  
female(Z),  
write(Z,"\n"),  
fail.
```

```
uncle(X):- brother(Z,Y),  
father(Z,X),  
male(Y),  
write(Y,"\\n"),  
fail.
```

```
aunt(X):- husband(Z,Y),  
brother(Z,V),  
father(V,X),  
write(Y,"\\n"),  
fail.
```

```
cousin(X):- father(Z,X),  
father(V,Y),  
Z<>V,  
brother(V,Z),  
write(Y,"\\n").
```

```
repeat.
```

```
repeat:- repeat.
```

```
action(1):- write("\\nEnter name of person whose father is to be found : "),  
readln(X),  
write("\\n"),  
write("Father of ",X," is:"),  
father(Z,X),  
write(Z,"\\n"),  
fail.
```

```
action(2):- write("\\nEnter name of person whose mother is to be found : "),  
readln(X),  
write("\\n"),  
write("Mother of ",X," is:"),  
mother(Z,X),
```

write(Z,"\\n"),

fail.

action(3):- write("\\nEnter name of person whose brothers are to be found : "),

readln(X),

write("\\n"),

write("Brothers of ",X," are:\\n"),

listbrothers(X),

write("\\n"),

fail.

action(4):- write("\\nEnter name of person whose sisters are to be found : "),

readln(X),

write("\\n"),

write("Sisters of ",X," are:\\n"),

listsisters(X),

write("\\n"),

fail.

action(5):- write("\\nEnter name of person whose grandsons are to be found : "),

readln(X),

write("\\n"),

write("Grandsons of ",X," are:\\n"),

listgrandsons(X),

write("\\n"),

fail.

action(6):- write("\\nEnter name of person whose granddaughters are to be found : "),

readln(X),

write("\\n"),

write("Granddaughters of ",X," are:\\n"),

listgranddaughters(X),

write("\\n"),

fail.

action(7):- write("\nEnter name of person whose uncles are to be found : "),

readln(X),

write("\n"),

write("Uncles of ",X," are:\n"),

uncle(X),

write("\n"),

fail.

action(8):- write("\nEnter name of person whose aunties are to be found : "),

readln(X),

write("\n"),

write("Aunties of ",X," are:\n"),

aunt(X),

write("\n"),

fail.

action(9):- write("\nEnter name of person whose cousins are to be found : "),

readln(X),

write("\n"),

write("Cousins of ",X," are:\n"),

cousin(X),

write("\n"),

fail.

action(0).

printmenu :-

repeat,

write("\n1. Display Father of?\n"),

write("2. Display Mother of?\n"),

write("3. List all brothers of?\n"),

write("4. List all sisters of?\n"),


```
write("5. List all grandson of?\n"),
write("6. List all granddaughter of?\n"),
write("7. List all uncles of?\n"),
write("8. List all aunty of?\n"),
write("9. list all cousins of?\n"),
write("0. exit\n"),
write("Enter your choice : "),
readInt(Choice),
action(Choice),
write("\n"),
repeat.
goal
makewindow(1,2,3,"Family Tree",0,0,25,80),
printmenu.
```

Output:-

+-----Family Tree ----- +

||

|1. Display Father of? |

|2. Display Mother of? |

|3. List all brothers of? |

|4. List all sisters of? |

|5. List all grandson of? |

|6. List all granddaughter of? |

|7. List all uncles of? |

|8. List all aunty of? |

|9. list all cousins of? |

|0. exit |

|Enter your choice : 1 |

||

|Enter name of person whose father is to be found : ram |

||

|Father of ram is:dashrath |

||

|1. Display Father of? |

|2. Display Mother of? |

|3. List all brothers of? |

|4. List all sisters of? |

|5. List all grandson of? |

|6. List all granddaughter of? |

|7. List all uncles of? |

|8. List all aunty of? |

|9. list all cousins of? |

|0. exit |

|Enter your choice : 3 |

||

|Enter name of person whose brothers are to be found : ram |

||

|Brothers of ram are: |

|laxman |

|bharat |

||

|1. Display Father of? |

|2. Display Mother of? |

|3. List all brothers of? |

|4. List all sisters of? |

|5. List all grandson of? |

|6. List all granddaughter of? |

|7. List all uncles of? |

|8. List all aunty of? |

|9. list all cousins of? |

|0. exit |

|Enter your choice : 5 |

||

|Enter name of person whose grandsons are to be found : dashrath |

||

|Grandsons of dashrath are: |

|luv |

|kush |

|son_of_laxman |

||

|1. Display Father of? |

|2. Display Mother of? |

|3. List all brothers of? |

|4. List all sisters of? |

|5. List all grandson of? |

|6. List all granddaughter of? |

|7. List all uncles of? |

|8. List all aunty of? |

|9. list all cousins of? |

|0. exit |

|Enter your choice : 7 |

||

|Enter name of person whose uncles are to be found : kus |

||

|Uncles of kush are: |

|laxman |

|bharat |

||

|1. Display Father of? |

|2. Display Mother of? |

|3. List all brothers of? |

|4. List all sisters of? |

|5. List all grandson of? |

|6. List all granddaughter of? |

|7. List all uncles of? |

|8. List all aunty of? |

|9. list all cousins of? |

|0. exit |

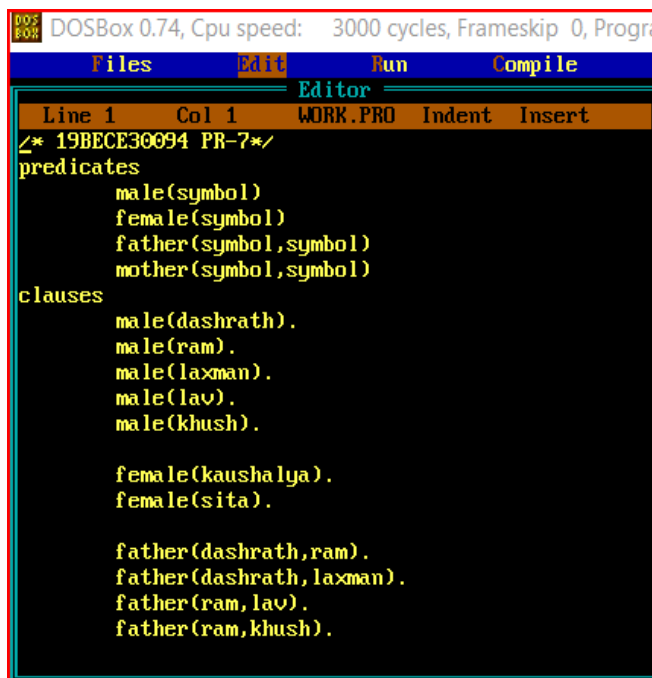
|Enter your choice : |

||

||

|Press the SPACE bar |

+-----+

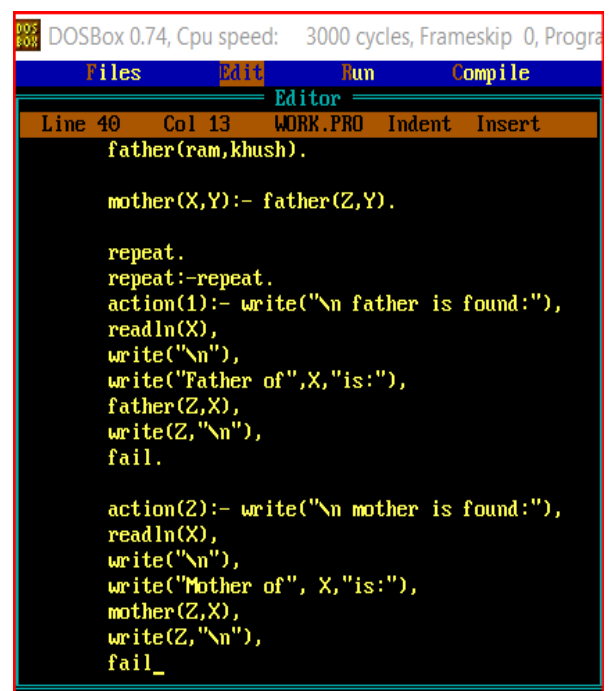


```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program WORK.PRO
Files Edit Run Compile
Line 1 Col 1 WORK.PRO Indent Insert
/* 19BECE30094 PR-7*/
predicates
    male(symbol)
    female(symbol)
    father(symbol,symbol)
    mother(symbol,symbol)
clauses
    male(dashrath).
    male(ram).
    male(laxman).
    male(lav).
    male(khush).

    female(kaushalya).
    female(sita).

    father(dashrath,ram).
    father(dashrath,laxman).
    father(ram,lav).
    father(ram,khush).
  
```



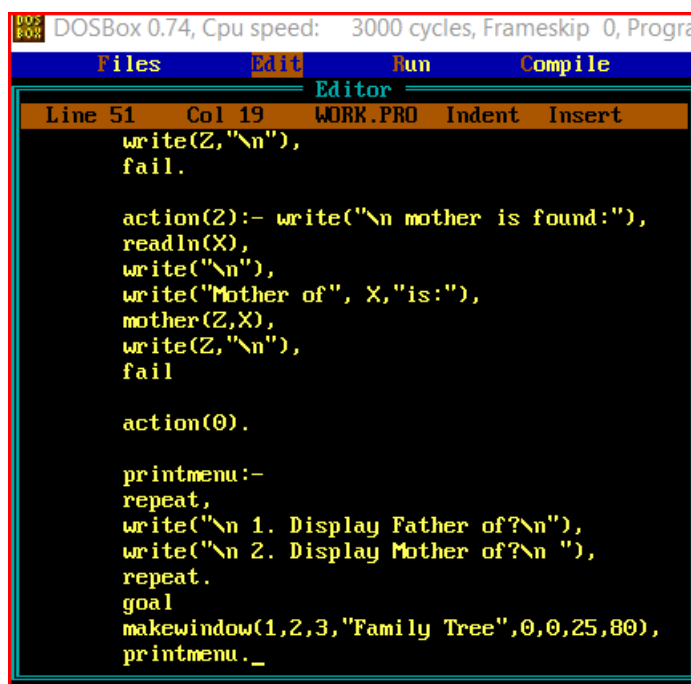
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program WORK.PRO
Files Edit Run Compile
Line 40 Col 13 WORK.PRO Indent Insert
    father(ram,khush).

    mother(X,Y):- father(Z,Y).

    repeat.
    repeat:-repeat.
    action(1):- write("\n father is found:"),
    readln(X),
    write("\n"),
    write("Father of",X,"is:"),
    father(Z,X),
    write(Z,"\n"),
    fail.

    action(2):- write("\n mother is found:"),
    readln(X),
    write("\n"),
    write("Mother of", X,"is:"),
    mother(Z,X),
    write(Z,"\n"),
    fail_
  
```



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program WORK.PRO
Files Edit Run Compile
Line 51 Col 19 WORK.PRO Indent Insert
    write(Z,"\n"),
    fail.

    action(2):- write("\n mother is found:"),
    readln(X),
    write("\n"),
    write("Mother of", X,"is:"),
    mother(Z,X),
    write(Z,"\n"),
    fail

    action(0).

    printmenu:-
    repeat,
    write("\n 1. Display Father of?\n"),
    write("\n 2. Display Mother of?\n "),
    repeat.
    goal
    makewindow(1,2,3,"Family Tree",0,0,25,80),
    printmenu._
  
```

Practical-8

Program : Write a program to implement BFS (for AI search problem)

domains

X, H, N, ND=symbol

P, L, T, Z, Z1, L1, L2, L3, PS, NP, ST, SOL=symbol*

predicates

solve(L, L) member(X,L) extend(L, L) conc(X, L, L) breadthfirst(L, L) goal(X)

clauses

solve(start, solution):-/*solution is a state from start to a goal*/ breadthfirst ([[start]],
solution).

breadthfirst([[node|path]| _],[node|path]):- /*solution is an extension to a
goal*/ /*of one of path*/

goal(node).

breadthfirst(path1,solution).

extend([node|path],newpaths):- bagof([newnode, node|path],(s(node,
newnode),notmember(newnode,[node|path])), newpaths),!.
extend(path, []). conc([], L, L).

conc([X|L1], L2, [X|L3]):- conc(L1, L2, L3).

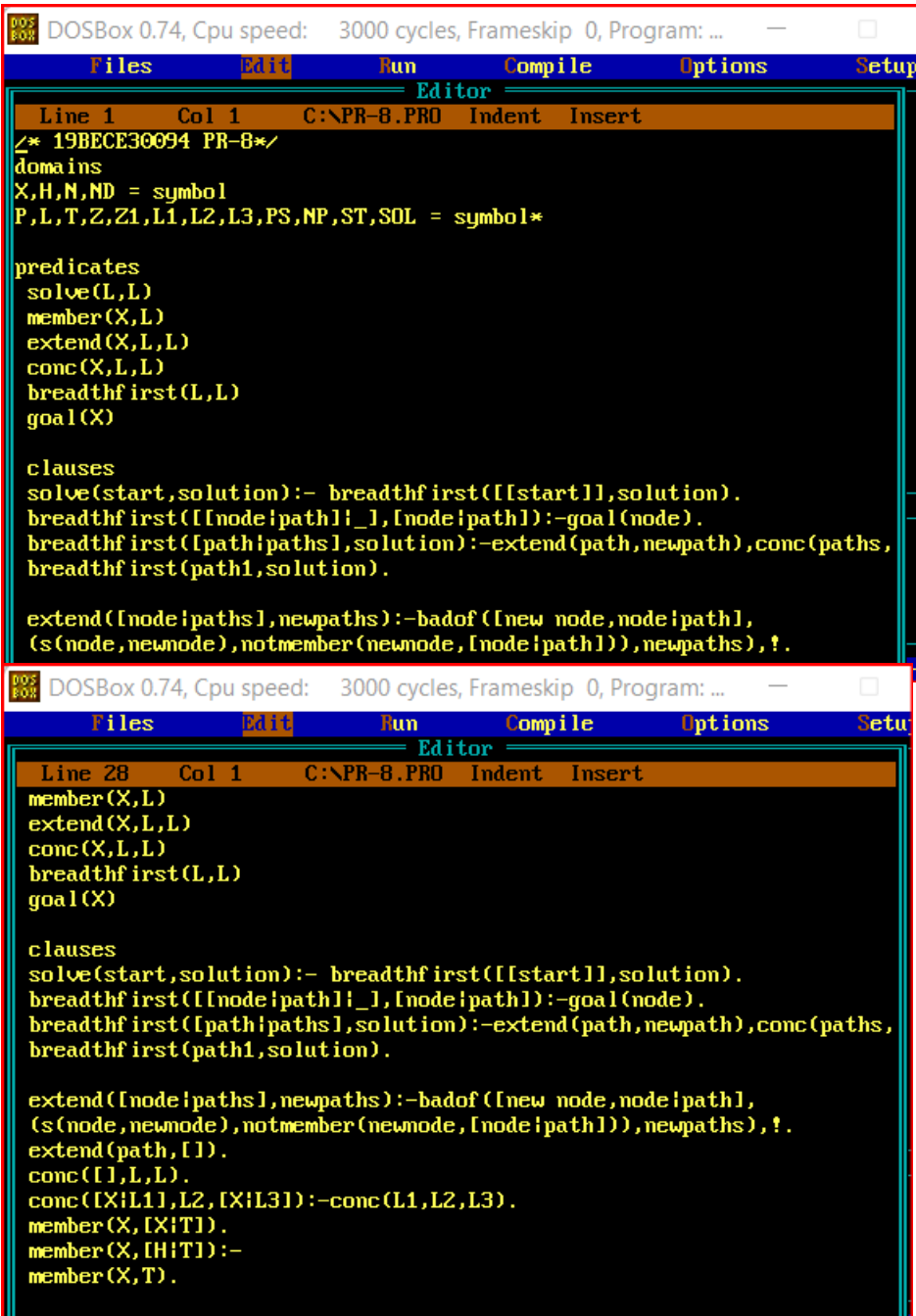
member(X, [X|T]).

member(X, [H|T]):-
member(X, T).

OUTPUT:-

```
goal: solve([a, e], S)
L= ["a", "b", "c", "d", "e"]

goal: solve([a, h],S)
L= ["a", "b", "c", "d", "e", "f", "g", "h"]
```



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...

Files Edit Run Compile Options Setup

Editor

Line 1 Col 1 C:\PR-8.PRO Indent Insert

```

/* 19BECE30094 PR-8*/
domains
X,H,N,ND = symbol
P,L,T,Z,Z1,L1,L2,L3,PS,NP,ST,SOL = symbol*

predicates
solve(L,L)
member(X,L)
extend(X,L,L)
conc(X,L,L)
breadthfirst(L,L)
goal(X)

clauses
solve(start,solution):- breadthfirst([[start]],solution).
breadthfirst([[_node:path]_],[_node:path]):-goal(node).
breadthfirst([path:paths],solution):-extend(path,newpath),conc(paths,
breadthfirst(path1,solution).

extend([_node:paths],newpaths):-badof([new_node,node:path],
(s(node,newnode),notmember(newnode,[_node:path])),newpaths),?.

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...

Files Edit Run Compile Options Setup

Editor

Line 28 Col 1 C:\PR-8.PRO Indent Insert

```

member(X,L)
extend(X,L,L)
conc(X,L,L)
breadthfirst(L,L)
goal(X)

clauses
solve(start,solution):- breadthfirst([[start]],solution).
breadthfirst([[_node:path]_],[_node:path]):-goal(node).
breadthfirst([path:paths],solution):-extend(path,newpath),conc(paths,
breadthfirst(path1,solution).

extend([_node:paths],newpaths):-badof([new_node,node:path],
(s(node,newnode),notmember(newnode,[_node:path])),newpaths),?.
extend(path, []).
conc([],L,L).
conc([X:L1],L2,[X:L3]):-conc(L1,L2,L3).
member(X,[X:T]).
member(X,[_H:T]):-
member(X,T).

```

Practical : 9**Program : Write a program to implement DFS (for 8 puzzle problem)**

domains

H=integer T=integer*

predicates

safe(T)

solution(T)

permutation(T,T)

del(H,T,T)

noattack(H,T,H)

clauses

```
del(I,[I|L],L). /*to take a position from the permutation of
list*/ del(I,[F|L],[F|L1]):-
del(I,L,L1).
```

```
permutation([],[]). /*to find the possible positions*/
permutation([H|T],PL):-
permutation(T,PT),\ del(H,PL,PT).
```

```
solution(Q):- /*final solution is stored in Q*/
permutation([1,2,3,4,5,6,7,8],Q),
safe(Q).
```

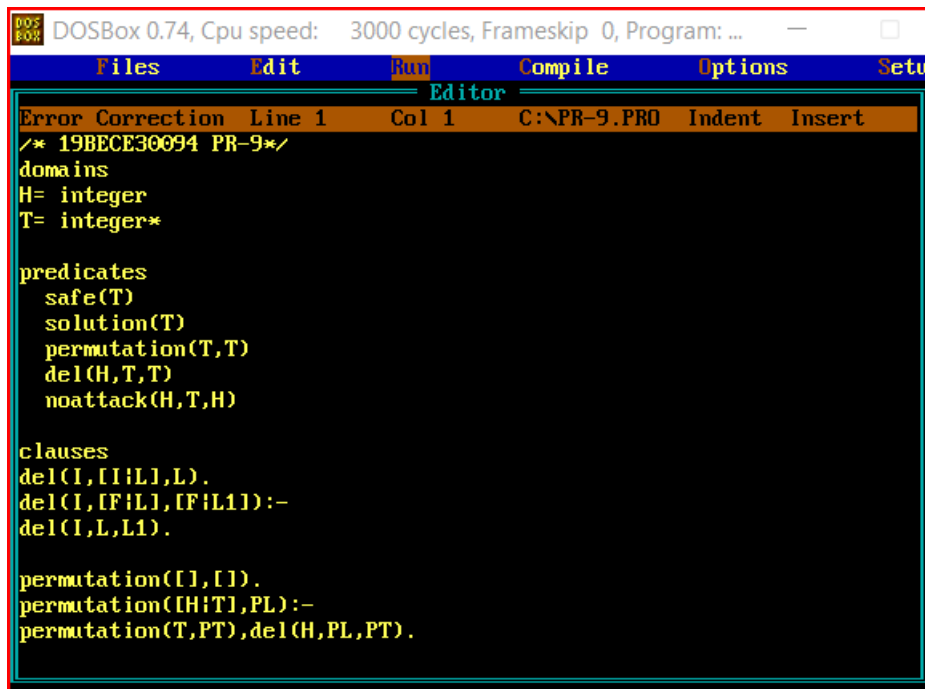
```
safe([]). /*Q is safe such that no queens attack each other*/
safe([Q|others]):-
safe(others), noattack(Q,others,1).
```

```
noattack(_,[],_)./*to find if the queens are in same row, column or
diagonal*/
```

```
noattack(Y,[Y1|Ydist],Xdist):-
Y1-Y >Xdist,
Y-Y1 <>Xdist,
dist1=Xdist,
noattack(Y,Ydist,dist1)
```

OUTPUT:-

```
goal:-solution(Q). Q=["3","8","4","7","1","6","2","5"]
```

```

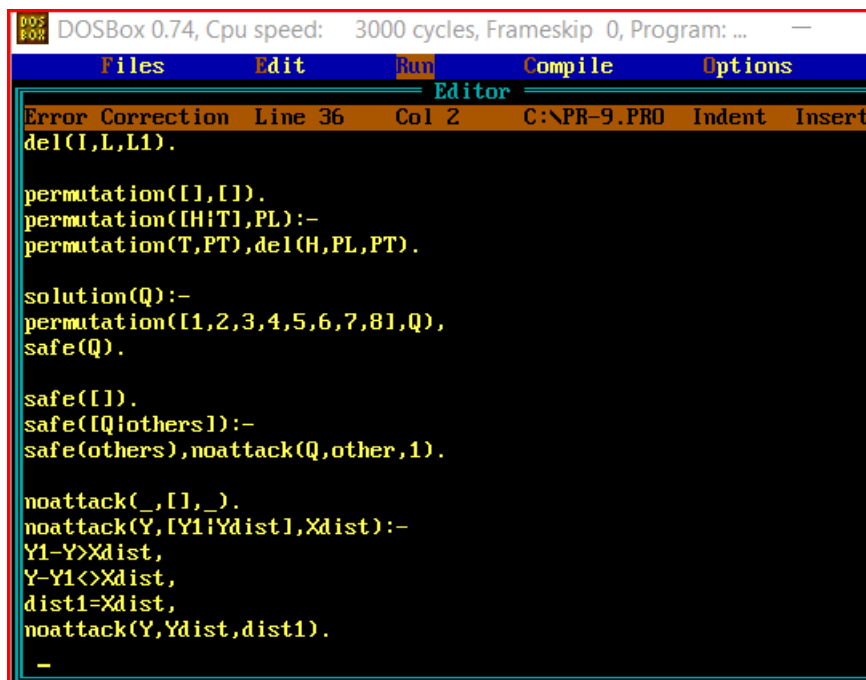
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setu
Error Correction Line 1 Col 1 C:\PR-9.PRO Indent Insert
/* 19BECE30094 PR-9*/
domains
H= integer
T= integer*

predicates
  safe(T)
  solution(T)
  permutation(T,T)
  del(H,T,T)
  noattack(H,T,H)

clauses
del(I,[I:L1],L).
del(I,[F:I1],F:I1):-
del(I,L,L1).

permutation([],[]).
permutation([H:T1],PL):-
permutation(T,PT),del(H,PL,PT).

```



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options
Error Correction Line 36 Col 2 C:\PR-9.PRO Indent Insert
del(I,L,L1).

permutation([],[]).
permutation([H:T1],PL):-
permutation(T,PT),del(H,PL,PT).

solution(Q):-
permutation([1,2,3,4,5,6,7,8],Q),
safe(Q).

safe([]).
safe([Q:others1]):-
safe(others1),noattack(Q,other,1).

noattack(_,[],_).
noattack(Y,[Y1:Ydist],Xdist):-
Y1-Y>Xdist,
Y-Y1<>Xdist,
dist1=Xdist,
noattack(Y,Ydist,dist1).
-

```

Practical-10

Program : Write a program to implement A* Algorithm.

```

%%%
%%%
%%% Nodes have form S#D#F#A
%%% where S describes the state or

configuration
%%% D is the depth of the node
%%% F is the evaluation function value
%%% A is the ancestor list for the node

:- op(400,yfx,'#'). /* Node builder notation */

solve(State,Soln) :- f_function(State,0,F),
                    search([State#0#F#[]],S), reverse(S,Soln).

f_function(State,D,F) :- h_function(State,H),
                        F is D + H.

search([State#_#_#Soln|_], Soln) :- goal(State).
search([B|R],S) :- expand(B,Children),
                  insert_all(Children,R,Open),
                  search(Open,S).
insert_all([F|R],Open1,Open3) :- insert

(F,Open1,Open2),
                  insert_all(R,Open2,Open3).
insert_all([],Open,Open).

insert(B,Open,Open) :- repeat_node(B,Open), ! .
insert(B,[C|R],[B,C|R]) :- cheaper(B,C), ! .
insert(B,[B1|R],[B1|S]) :- insert(B,R,S), ! .
insert(B,[],[B]).

repeat_node(P#_#_#_, [P#_#_#_|_]).

cheaper(_#_#F1#_ , _#_#F2#_ ) :- F1 < F2.

expand(State#D#_#S,All_My_Children) :-
    bagof(Child#D1#F#[Move|S],
        (D1 is D+1,
         move(State,Child,Move),
         f_function(Child,D1,F)),
        All_My_Children).

```

Output:-

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...

Files Edit Run Compile Options Setup

Editor

Line 1 Col 1 WORK.PRO Indent Insert

```

/* 19BECE30094 PR-10*/
%%%
%%%
%%% Node have form S#D#F#A
%%% where S describes the the state or configuration
%%% D is the depth of the node
%%% F is the evaluation function value
%%% A is the ancestor list for the node
:- op(400,yfx,'#').
solve(State,Slon):-f_function(State,0,F),
search([State#0#f#[]],S),reverse(S,Soln).

f_function(State,D,F):-h_function(State,H),F is D+H.
  
```

Message

Load WORK.PRO
Save WORK.PRO
Compiling WORK.PRO

Trace

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-Xedit F9-Compile F10-Menu

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...

Files Edit Run Compile Options Setup

Editor

Line 14 Col 1 WORK.PRO Indent Insert

```

f_function(State,D,F):-h_function(State,H),F is D+H.

search([State#_#_#Slon!_],Slon):-goal(State).
search([B:R1],S):-expand(B,Children),
insert_all(Children,R,Open),
search(Open,S).

insert_all([F:R1],Open1,Open3):-insert(F,Open1,Open2)
insert_all(R,Open2,Open3).
insert_all([],Open,Open).

insert(B,Open,Open):-repeat_node(B,Open),!.
insert(B,[C:R1],[B,C:R1):-cheaper(B,C),!.
insert(B,[B1:R1],[B1:S1):-insert(B,R,S),!.
  
```

Message

Load WORK.PRO
Save WORK.PRO
Compiling WORK.PRO

Trace

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-Xedit F9-Compile F10-Menu

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program:

Files Edit Run Compile Op

Editor

Line 35 Col 18 WORK.PRO Indent Insert

```

insert_all([],Open,Open).

insert(B,Open,Open):-repeat_node(B,Open),!.
insert(B,[C|R],[B,C|R]):-cheaper(B,C),!.
insert(B,[B1|R],[B1|S]):-insert(B,R,S),!.
insert(B,[],[B]).

repeat_node(P#_#_#_,[P#_#_#_!_]).
cheaper(_#_#F1#_#_#F2#_):-F1<F2.

expand(SSState#D#_#S,All_My_Children):-
bagof(Child#D1#F#[Move!S1,(D1 is D+1,move(State,Child,
f_function(Child,D1,F)),
All_My_Children).
  
```

Message

Load WORK.PRO
Save WORK.PRO
Compiling WORK.PRO

F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Xcopy F8-Xedi

Practical-11

Program : Write a program to solve travelling salesman problem using Prolog.

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).

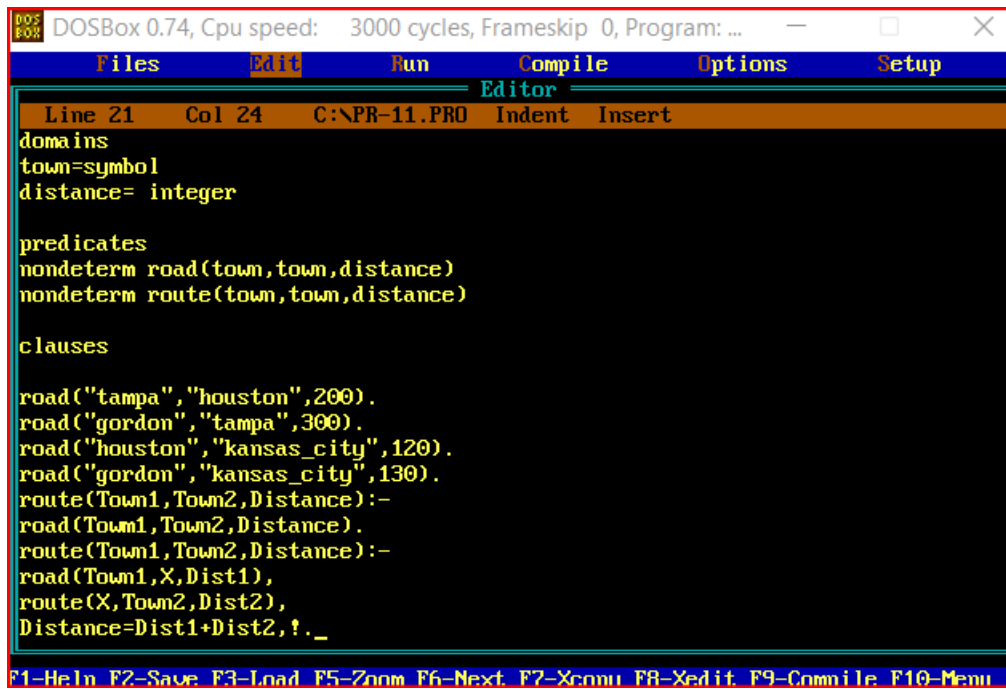
route(Town1, Town2,Distance):- road(Town1,X, Dist1), route(X,Town2,Dist2),
Distance=Dist1+Dist2, !.

```
/* 19BECE30094 PR-11*/
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","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),
```



The screenshot shows the DOSBox 0.74 interface. The title bar indicates 'DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...'. The menu bar includes 'Files', 'Edit', 'Run', 'Compile', 'Options', and 'Setup'. The 'Editor' window is active, displaying a Prolog program. The status bar at the bottom shows keyboard shortcuts: 'F1-Help F2-Save F3-Load F5-Zoom F6-Next F7-Exit F8-Xedit F9-Compile F10-Menu'.

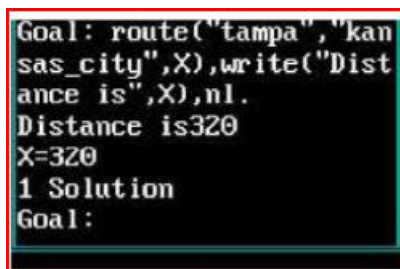
```
Line 21 Col 24 C:\PR-11.PRO Indent Insert
domains
town= symbol
distance= integer

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

clauses

road("tampa","houston",200).
road("gordon","tampa",300).
road("houston","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:-



The screenshot shows the output of the Prolog program. It displays the goal being executed, the distance calculated, and the solution found.

```
Goal: route("tampa","kansas_city",X),write("Distance is",X),nl.
Distance is 320
X=320
1 Solution
Goal:
```

PRACTICAL : 12

Program : Study of dynamic database in PROLOG.

Predicates

reading

writing

delete

find(integer)

startup(integer)

Database

unsortedDatabase(string, integer)

sortedDatabase(string)

clauses

startup(0).

startup(Num):-

write("Enter String = "),

readln(Name),

str_len(Name,Len),

asserta(unsortedDatabase(Name, Len)),

TempNum = Num - 1,

startup(TempNum).

writing:

sortedDatabase(Name),

write(Name),nl,

fail.

writing.

find(Index):

unsortedDatabase(Name,Index),

assertz(sortedDatabase(Name)),

retract(unsortedDatabase(Name,Index)),

find(Index).

find(Index):-

Index = 255.

find(Index):-

TemplIndex = Index + 1,

find(TemplIndex).

reading:-

NumRead = 10,

startup(NumRead).

delete :

retract(sortedDatabase(_)),

fail.

delete.

Goal

Clearwindow,

makewindow(1,2,3,"String Operations",0,0,25,80),

reading,!,

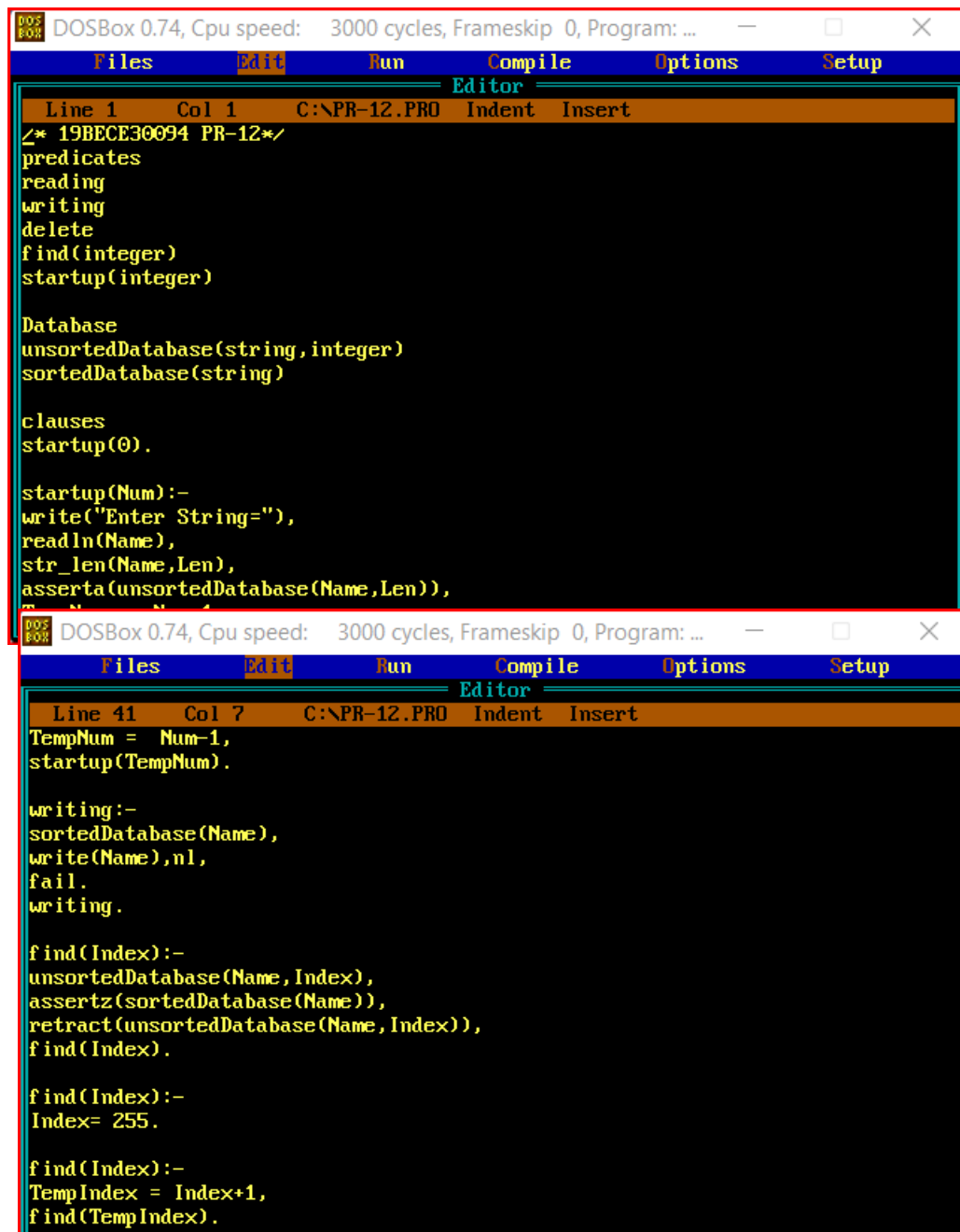
find(1),

write("\nString In Increasing Order Of Their Length Are : \n"),

writing,

delete.

OutPut:-



The image displays two screenshots of a DOSBox 0.74 window, showing a Prolog program in an editor. The window title is "DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...". The menu bar includes "Files", "Edit", "Run", "Compile", "Options", and "Setup". The status bar shows "Line 1 Col 1 C:\PR-12.PRO Indent Insert".

The first screenshot shows the following code:

```

/* 19BECE30094 PR-12*/
predicates
reading
writing
delete
find(integer)
startup(integer)

Database
unsortedDatabase(string, integer)
sortedDatabase(string)

clauses
startup(0).

startup(Num):-
write("Enter String="),
readln(Name),
str_len(Name, Len),
asserta(unsortedDatabase(Name, Len)),

```

The second screenshot shows the following code:

```

TempNum = Num-1,
startup(TempNum).

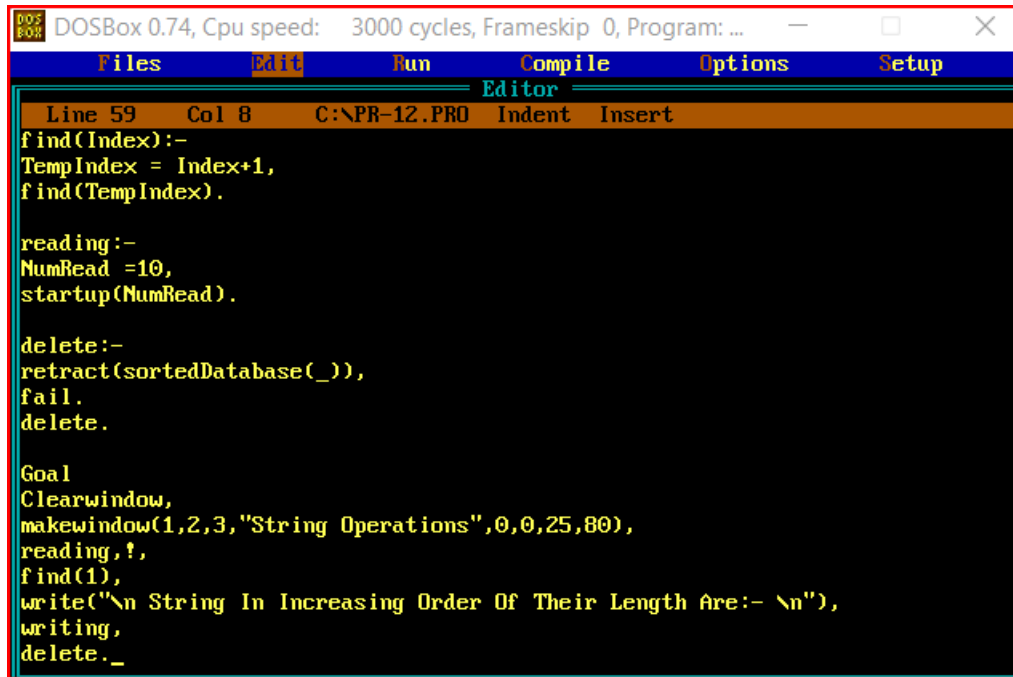
writing:-
sortedDatabase(Name),
write(Name), nl,
fail.
writing.

find(Index):-
unsortedDatabase(Name, Index),
assertz(sortedDatabase(Name)),
retract(unsortedDatabase(Name, Index)),
find(Index).

find(Index):-
Index= 255.

find(Index):-
TempIndex = Index+1,
find(TempIndex).

```

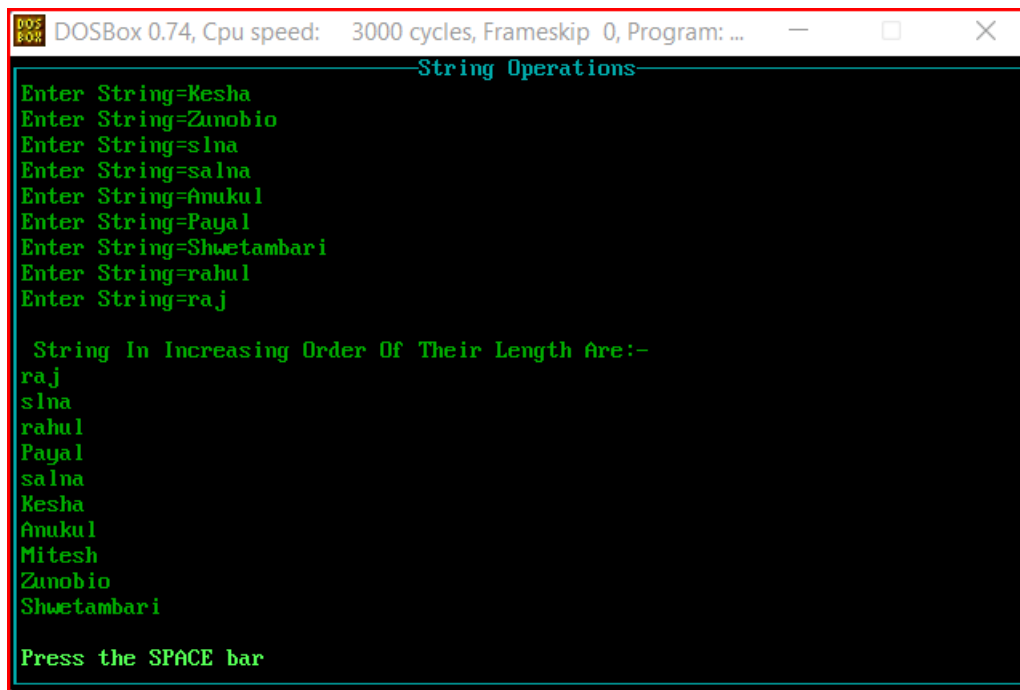


```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Files Edit Run Compile Options Setup
Editor
Line 59 Col 8 C:\PR-12.PRO Indent Insert
find(Index):-
TempIndex = Index+1,
find(TempIndex).

reading:-
NumRead =10,
startup(NumRead).

delete:-
retract(sortedDatabase(_)),
fail.
delete.

Goal
Clearwindow,
makewindow(1,2,3,"String Operations",0,0,25,80),
reading,!,
find(1),
write("\n String In Increasing Order Of Their Length Are:- \n"),
writing,
delete._
```



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
String Operations
Enter String=Kesha
Enter String=Zunobio
Enter String=slna
Enter String=salna
Enter String=Anukul
Enter String=Payal
Enter String=Shwetambari
Enter String=rahul
Enter String=raj

String In Increasing Order Of Their Length Are:-
raj
slna
rahul
Payal
salna
Kesha
Anukul
Mitesh
Zunobio
Shwetambari

Press the SPACE bar
```