

PORT CITY INTERNATIONAL UNIVERSITY

Dept. of Computer Science & Engineering

REPORT ON

Solution of the given problems

COURSE TITLE : Artificial Intelligent Sessional

COURSE CODE : CSE 342

DATE OF SUBMISSION: 10.04.2021

SUBMITTED TO

Taslima Binte Hossain

lecturer of CSE dept. PCIU

SUBMITTED BY-

NAME: Saimon akram

ID: CSE 013 06167

BATCH: 13TH B

SESSION: Spring,2021

REMARKS

Problem No: 01

Problem Name: Create a medical diagnosis expert system using prolog

Objective: To implement a medical expert system which can diagnose disease.

Problem Statement: Expert systems are considered a subset or an application of the branch of computer science known as artificial intelligence. In turn, artificial intelligence is broadly defined as comprising certain techniques that allow computers to take on the characteristics of human intelligence

A medical expert system is a computer program that, when well-crafted, gives decision support in the form of accurate diagnostic information or, less commonly, suggests treatment or prognosis. Diagnostic, therapeutic, or prognostic advice is given after the program receives information (input) about the patient, usually via the patient's physician. Expert systems have characteristics which make them dissimilar from other kinds of medical software. One of these characteristics is that the sequence of steps used by the expert system in coming to a diagnostic or therapeutic conclusion often is designed to mimic clinical reasoning. Also, the sequence of steps is, in many expert systems, available to the physician using the system. Because clinical medicine often does not deal in certainty, expert systems may have the capability of expressing conclusions as a probability. It is generally agreed that expert system software must contain a large number of facts and rules about the disease or condition in question in order to deliver accurate answers. It has been estimated that two general internal medicine textbooks and three specialty textbooks would require 2 million rules. Because large amounts of data are needed, in the recent past, expert systems were only feasible when used with large, expensive computers. With the advent of more powerful microcomputers and more efficient microcomputer languages, expert systems could now be available to any physician with a microcomputer.

Source Code:

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: PROLOG
Files Edit Run Compile Options Setup
Line 1 Col 1 E:\MEDICAL.PRO Indent Insert
domains
    patient,dis,indication=symbol

predicates
    go
    hypothesis(patient,dis)
    symptom(patient,indication)
    response(char)
    repeat
    go_once

clauses
    go:-
        repeat,
        go_once,
        write("would you like to try again?(y/n)"),
        response(Reply),
        Reply='n'.
    go.
    repeat.
    repeat:-
        repeat.

    go_once:-
        write("What is the patient's Name?"),
        readln(Patient),
        hypothesis(Patient,Dis),
        write(Patient," probably has ",Dis,"."),nl.

    go_once:-
        write("Sorry i dont understand to be able to diagnose the disease.")

    hypothesis(Patient,measles):-
        symptom(Patient,fever),
        symptom(Patient,cough),
        symptom(Patient,conjunctivitis),
        symptom(Patient,runny_nose),
        symptom(Patient,rash).

    hypothesis(Patient,german_measles):-
        symptom(Patient,fever),
        symptom(Patient,headache),
        symptom(Patient,runny_nose).
```

```

hypothesis(Patient,flu):-
    symptom(Patient,fever),
    symptom(Patient,headache),
        symptom(Patient,body_ache),
    symptom(Patient,conjunctivitis),
        symptom(Patient,chillis),
    symptom(Patient,runny_nose),
        symptom(Patient,rash).

symptom(Patient,fever):-
    write("Does ",Patient,"have a fever(Y/N)?"),
    response(Reply),
    Reply='Y'.

symptom(Patient,cough):-
    write("Does ",Patient,"have a cough(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 runny_nose(Y/N)?"),
    response(Reply),
    Reply='Y'.

symptom(Patient,rash):-
    write("Does ",Patient,"have a rash(Y/N)?"),
    response(Reply),
    Reply='Y'.

symptom(Patient,chillis):-
    write("Does ",Patient,"have a chillis(Y/N)?"),
    response(Reply),
    Reply='Y'.

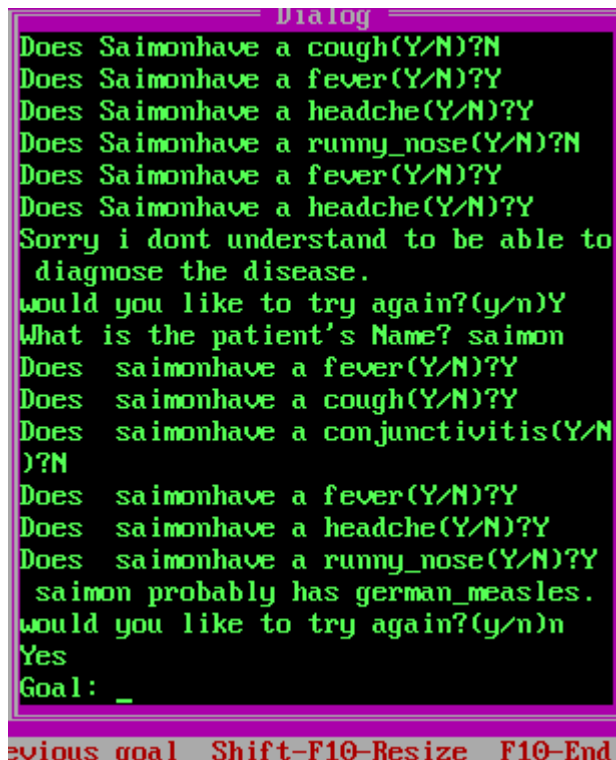
symptom(Patient,conjunctivitis):-
    write("Does ",Patient,"have a conjunctivitis(Y/N)?"),
    response(Reply),
    Reply='Y'.

symptom(Patient,body_ache):-
    write("Does ",Patient,"have a body ache(Y/N)?"),
    response(Reply),
    Reply='Y'.

response(Reply):-
    readchar(Reply),
    write(Reply),nl.

```

Output:



```
Dialog
Does Saimonhave a cough(Y/N)?N
Does Saimonhave a fever(Y/N)?Y
Does Saimonhave a headche(Y/N)?Y
Does Saimonhave a runny_nose(Y/N)?N
Does Saimonhave a fever(Y/N)?Y
Does Saimonhave a headche(Y/N)?Y
Sorry i dont understand to be able to
diagnose the disease.
would you like to try again?(y/n)Y
What is the patient's Name? saimon
Does saimonhave a fever(Y/N)?Y
Does saimonhave a cough(Y/N)?Y
Does saimonhave a conjunctivitis(Y/N
)?N
Does saimonhave a fever(Y/N)?Y
Does saimonhave a headche(Y/N)?Y
Does saimonhave a runny_nose(Y/N)?Y
saimon probably has german_measles.
would you like to try again?(y/n)n
Yes
Goal: _
Previous goal Shift-F10-Resize F10-End
```

Problem No: 02

Problem Name: Create a logon system by which a person can successfully login using user name and password in prolog.

Objective: To Implement a login system where user can input user name and password.

Problem Statement: Adding a new user login to the system- useradd · Change login password and associated attributes - passwd · Write a menu driven program to display message, user name ,user password authentication.

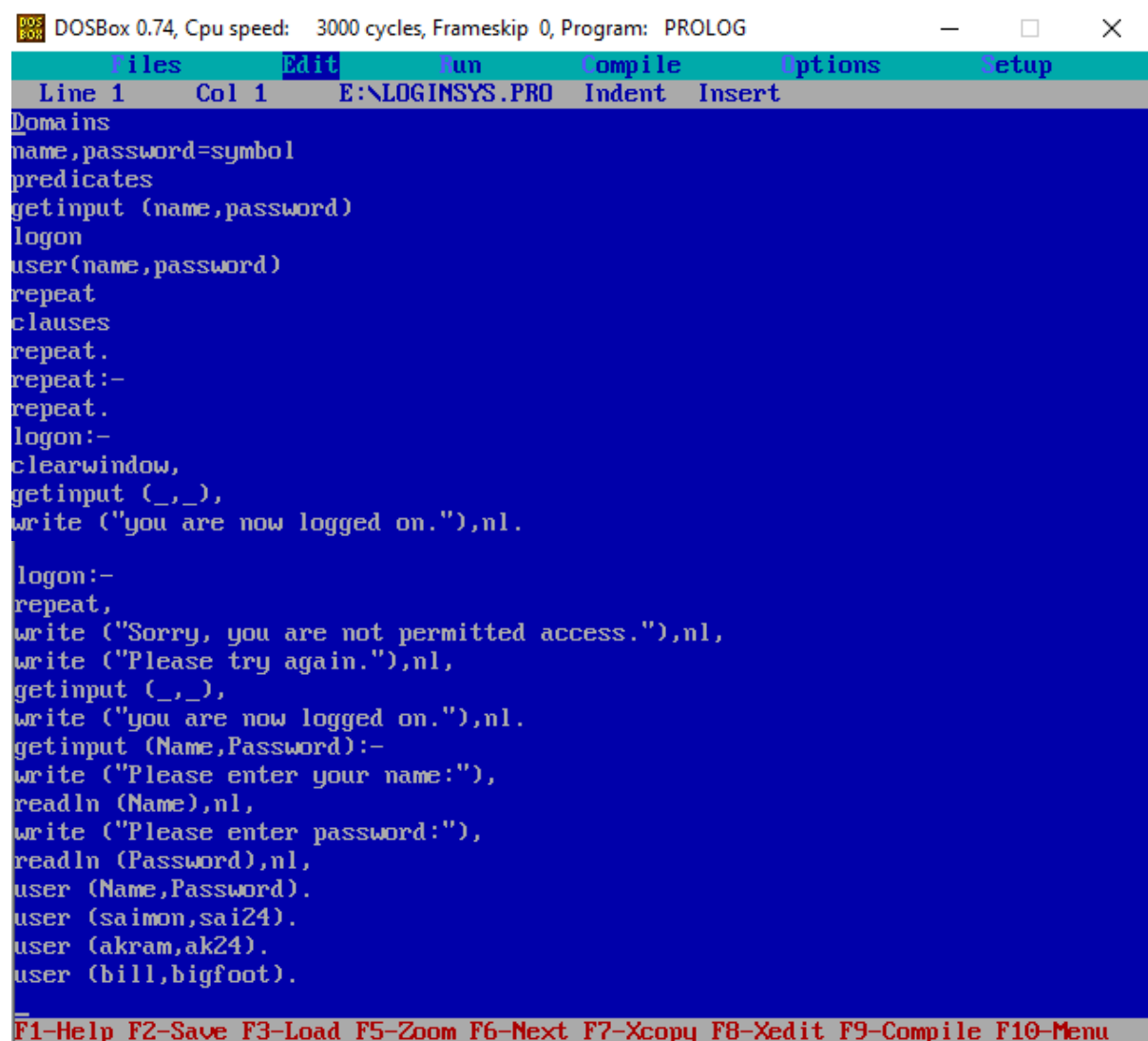
If the user name and password are matched

then it shows –“You are now logged on”

else the user name and password are not matched

then it shows-“Sorry,Your are not permitted access”.

Source Code:

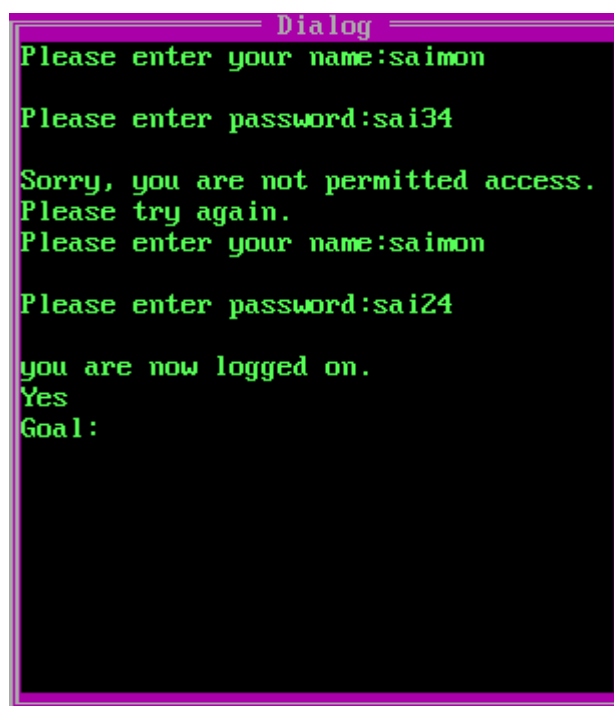


```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: PROLOG
Files Edit Run Compile Options Setup
Line 1 Col 1 E:\LOGINSYS.PRO Indent Insert
Domains
name,password=symbol
predicates
getinput (name,password)
logon
user(name,password)
repeat
clauses
repeat.
repeat:-
repeat.
logon:-
clearwindow,
getinput (_,_),
write ("you are now logged on."),nl.

logon:-
repeat,
write ("Sorry, you are not permitted access."),nl,
write ("Please try again."),nl,
getinput (_,_),
write ("you are now logged on."),nl.
getinput (Name>Password):-
write ("Please enter your name:"),
readln (Name),nl,
write ("Please enter password:"),
readln (Password),nl,
user (Name>Password).
user (saimon,sai24).
user (akram,ak24).
user (bill,bigfoot).
```

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

Output:



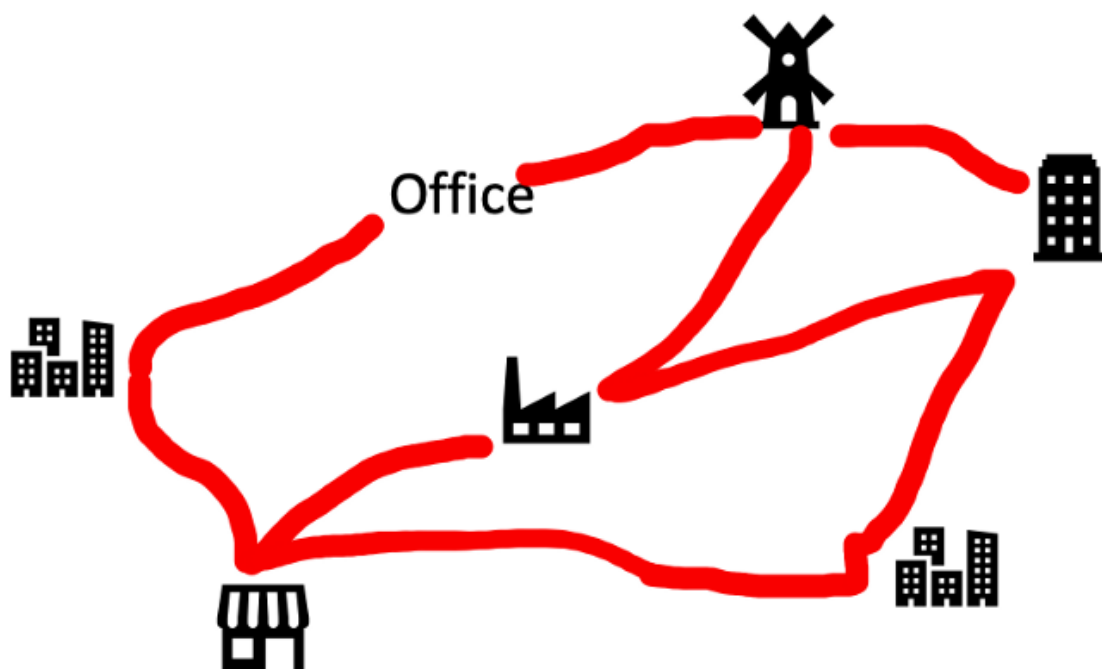
```
Dialog
Please enter your name:saimon
Please enter password:sai34
Sorry, you are not permitted access.
Please try again.
Please enter your name:saimon
Please enter password:sai24
you are now logged on.
Yes
Goal:
```

Problem No: 03

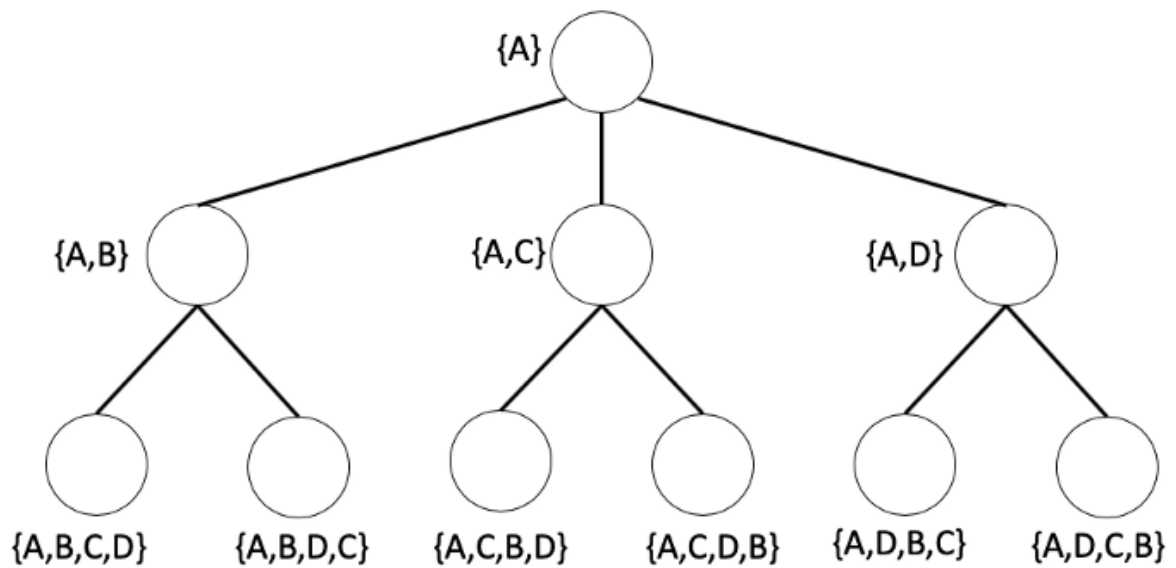
Problem Name: Implement Travelling salesman problem in prolog.

Objective: To implement the solution of TSP and chose shortest path in prolog

Problem Statement: Consider the following situation. You are given a list of n cities with the distance between any two cities. Now, you have to start with your office and to visit all the cities only once each and return to your office. What is the shortest path can you take? This problem is called the Traveling Salesman Problem (TSP).



To make the problem simple, we consider 3-city-problem. Let's call the office (A) and the 3 cities (B) (C) (D) respectively. We initialize the problem state by { A } means the salesman departed from his office. As an operator, when he visited city-B, the problem state is updated to { A, B }, where the order of elements in { } is considered. When the salesman visited all the cities, { A, B, C, D } in this case, the departed point A is automatically added to the state which means { A, B, C, D, A }. Therefore, the initial state of this TSP is { A } and the final state(goal) is { A, X1, X2, X3, A } where traveled distance is minimized. Taking each state as a node of a tree structure, we can represent this TSP as the following tree search problem.



Source Code:

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: PROLOG
Files Edit Run Compile Options Setup
Line 1 Col 1 E:\TSP.PRO Indent Insert
domains
town = symbol

distance = Integer

rib = r(town,town,distance)

tlist = town*

rlist = rib*

predicates

nondeterm way(town,town,rlist,distance)

nondeterm route(town,town,rlist,tlist,distance)

nondeterm route1(town,tlist,rlist,tlist,distance)

nondeterm ribsmember(rib,rlist)

nondeterm townsmember(town,tlist)

nondeterm tsp(town,town,tlist,rlist,tlist,distance)

nondeterm ham(town,town,tlist,rlist,tlist,distance)

nondeterm shorterRouteExists(town,town,tlist,rlist,distance)

nondeterm alltown(tlist,tlist)

nondeterm write_list(tlist)

```


clauses

```
write_list(L).
write_list([H:T]):-
    write(H,' '),
    write_list(T).

townsmember(X,[X:_]).
townsmember(X,[_:L]):-
    townsmember(X,L).

ribsmember(r(X,Y,D),[r(X,Y,D):_]).
ribsmember(X,[_:L]):-
    ribsmember(X,L).

alltown(_,[]).
alltown(Route,[H:T]):-
    townsmember(H,Route),
    alltown(Route,T).

way(Town1,Town2,Ways,OutWayDistance):-
    ribsmember(r(Town1,Town2,D),Ways),
    OutWayDistance = D.
/**

way(Town1,Town2,Ways,OutWayDistance):-
    ribsmember(r(Town2,Town1,D),Ways), /*switching direction here!G^a*/
    OutWayDistance = D.

**/

/* Is true if we could build route from Town1 to Town2 */
route(Town1,Town2,Ways,OutRoute,OutDistance):-
    route1(Town1,[Town2],Ways,OutRoute,T1T2Distance),
    /**
    SWITCH HERE
    OutDistance = T1T2Distance.
    route1(Town1,[Town1:Route1],_,[Town1:Route1],OutDistance):-
        OutDistance = 0.
    route1(Town1,[Town2:PassedRoute],Ways,OutRoute,OutDistance):-
        way(TownX,Town2,Ways,WayDistance),
        not(townsmember(TownX,PassedRoute)),

```

```

    route1(Town1,[TownX,Town2;PassedRoute],Ways,OutRoute,CompletingRoadDistance)

    OutDistance = CompletingRoadDistance + WayDistance.
shorterRouteExists(Town1,Town2,Towns,Ways,Distance):-
    ham(Town1,Town2,Towns,Ways,_,Other),
        Other < Distance.
tsp(Town1,Town1,Towns,Ways,BestRoute,MinDistance):-
    way(OtherTown,Town1,Ways,_),
        tsp(Town1,OtherTown,Towns,Ways,BestRoute,MinDistance).
-
tsp(Town1,Town2,Towns,Ways,BestRoute,MinDistance):-
    ham(Town1,Town2,Towns,Ways,Route,Distance),
        not(shorterRouteExists(Town1,Town2,Towns,Ways,Distance)),

    BestRoute = Route,

    MinDistance = Distance.
ham(Town1,Town2,Towns,Ways,Route,Distance):-
    route(Town1,Town2,Ways,Route,Distance),
%SWITCH HERE

    alltown(Route,Towns), % if you want simple road without including all towns
    write_list(Route),

    write(" tD = ",Distance,"\n").

% fail.
goal
/* EXAMPLE 1
    AllTowns = [a,b,c,d],
    AllWays = [r(a,b,1),r(a,c,10),r(c,b,2),r(b,c,2),r(b,d,5),r(c,d,3),r(d,a,4)],
*/
/* EXAMPLE 2 */
    AllTowns = [a,b,c,d,e],
    AllWays = [r(a,c,1),r(a,b,6),r(a,e,5),r(a,d,8),r(c,b,2),r(c,d,7),r(c,e,10),r(b
    tsp(a,a,AllTowns,AllWays,Route,Distance),

%SWITCH HERE

%   tsp(a,b,AllTowns,AllWays,Route,Distance),

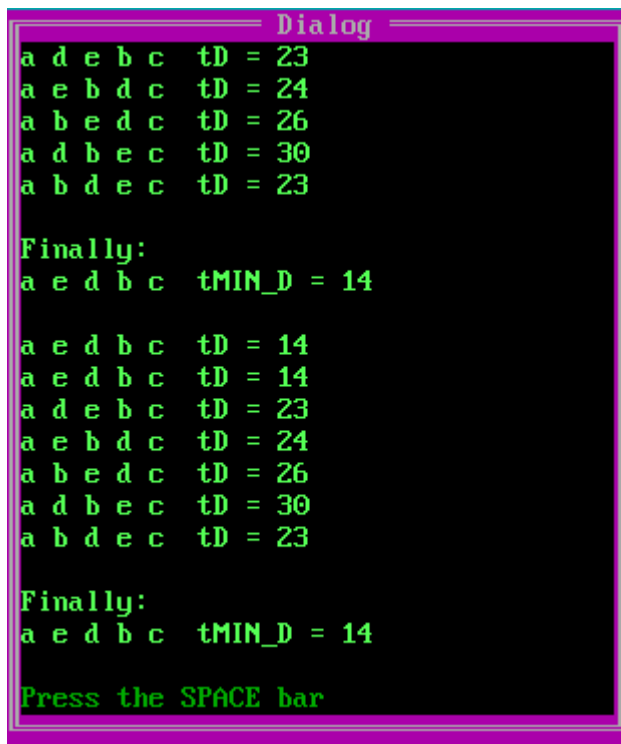
    write("\nFinally:\n"),

    write_list(Route),

    write(" tMIN_D = ",Distance,"\n").

```

Output:



```
Dialog
a d e b c tD = 23
a e b d c tD = 24
a b e d c tD = 26
a d b e c tD = 30
a b d e c tD = 23

Finally:
a e d b c tMIN_D = 14

a e d b c tD = 14
a e d b c tD = 14
a d e b c tD = 23
a e b d c tD = 24
a b e d c tD = 26
a d b e c tD = 30
a b d e c tD = 23

Finally:
a e d b c tMIN_D = 14

Press the SPACE bar
```

Discussion:

Through this lab, I learned about Turbo prolog, its basic structure and its application field .Our main purpose in this lab is to gathering skills declarative programming language that has an important role in artificial intelligence, and how it works with logical rules and structure. We learned about predicates, clausal relations, comparison operations, goals, queries, etc. And implemented the given problems.