Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Subject : Artificial Intelligence Lab

Name : Mohammad Anas Aarif Baig Mirza **Roll No. :** 40

Class : B.tech Final Year **Batch : B2** **Division: -**

Expt. No. :07 **Date : 16/10/2025**

Title : Solve traveling salesman problem.

Remark

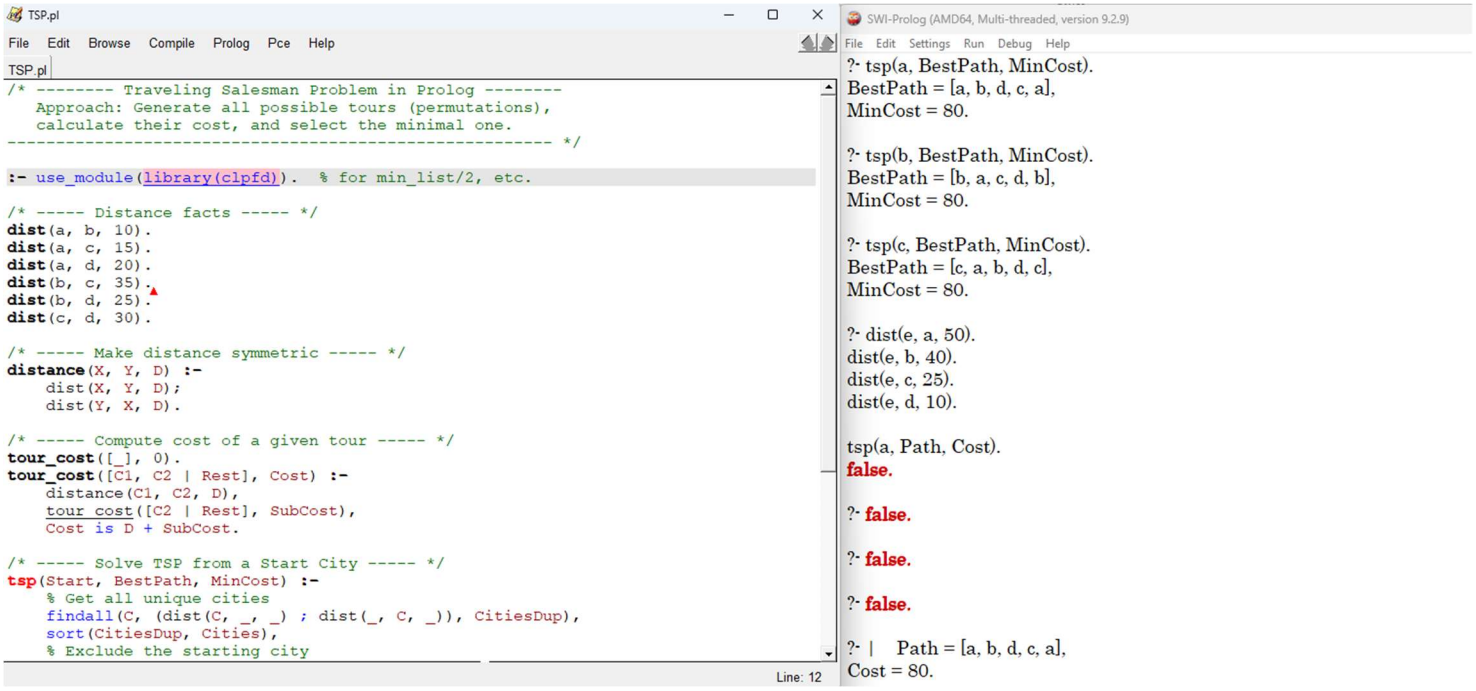
Signature

Program Code:

```
/* -------- Traveling Salesman Problem in Prolog --------
   Approach: Generate all possible tours (permutations),
   calculate their cost, and select the minimal one.
------------------------------------------------------- */
:- use_module(library(clpfd)).  % for min_list/2, etc.
/* ----- Distance facts ----- */
dist(a, b, 10).
dist(a, c, 15).
dist(a, d, 20).
dist(b, c, 35).
dist(b, d, 25).
dist(c, d, 30).
/* ----- Make distance symmetric ----- */
distance(X, Y, D) :-
    dist(X, Y, D);
    dist(Y, X, D).
```

```
/* ----- Compute cost of a given tour ----- */
tour_cost([_], 0).
tour_cost([C1, C2 | Rest], Cost) :-
    distance(C1, C2, D),
    tour_cost([C2 | Rest], SubCost),
    Cost is D + SubCost.
/* ----- Solve TSP from a Start City ----- */
tsp(Start, BestPath, MinCost) :-
    % Get all unique cities
    findall(C, (dist(C, _, _) ; dist(_, C, _)), CitiesDup),
    sort(CitiesDup, Cities),
    % Exclude the starting city
    delete(Cities, Start, OtherCities),
    % Generate all permutations of other cities
    findall(Path, permutation(OtherCities, Path), PermPaths),
    % Add Start at beginning and end of each path
    findall([Start | PWithEnd],
            (member(P, PermPaths),
             append(P, [Start], PWithEnd)),
            AllTours),
    % Calculate cost for each tour
    findall(Cost-Path,
            (member(Path, AllTours),
             tour_cost(Path, Cost)),
            CostedTours),
    % Sort by cost and choose the best one
    keysort(CostedTours, [MinCost-BestPath | _]).
```

Output:



TSP.pl

File Edit Browse Compile Prolog Pce Help

TSP.pl

```
/* -------- Traveling Salesman Problem in Prolog --------
   Approach: Generate all possible tours (permutations),
   calculate their cost, and select the minimal one.
------------------------------------------------------------ */

:- use_module(library(clpfd)).  % for min_list/2, etc.

/* ----- Distance facts ----- */
dist(a, b, 10).
dist(a, c, 15).
dist(a, d, 20).
dist(b, c, 35).
dist(b, d, 25).
dist(c, d, 30).

/* ----- Make distance symmetric ----- */
distance(X, Y, D) :-
    dist(X, Y, D);
    dist(Y, X, D).

/* ----- Compute cost of a given tour ----- */
tour_cost([_], 0).
tour_cost([C1, C2 | Rest], Cost) :-
    distance(C1, C2, D),
    tour_cost([C2 | Rest], SubCost),
    Cost is D + SubCost.

/* ----- Solve TSP from a Start City ----- */
tsp(Start, BestPath, MinCost) :-
    % Get all unique cities
    findall(C, (dist(C, _, _) ; dist(_, C, _)), CitiesDup),
    sort(CitiesDup, Cities),
    % Exclude the starting city
```

Line: 12

SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)

File Edit Settings Run Debug Help

```
?- tsp(a, BestPath, MinCost).
BestPath = [a, b, d, c, a],
MinCost = 80.

?- tsp(b, BestPath, MinCost).
BestPath = [b, a, c, d, b],
MinCost = 80.

?- tsp(c, BestPath, MinCost).
BestPath = [c, a, b, d, c],
MinCost = 80.

?- dist(e, a, 50).
dist(e, b, 40).
dist(e, c, 25).
dist(e, d, 10).

tsp(a, Path, Cost).
false.

?- false.

?- false.

?- false.

?- |    Path = [a, b, d, c, a],
Cost = 80.
```