TE Comps-A
Manasvi Patil – 9219

Experiment No: 7

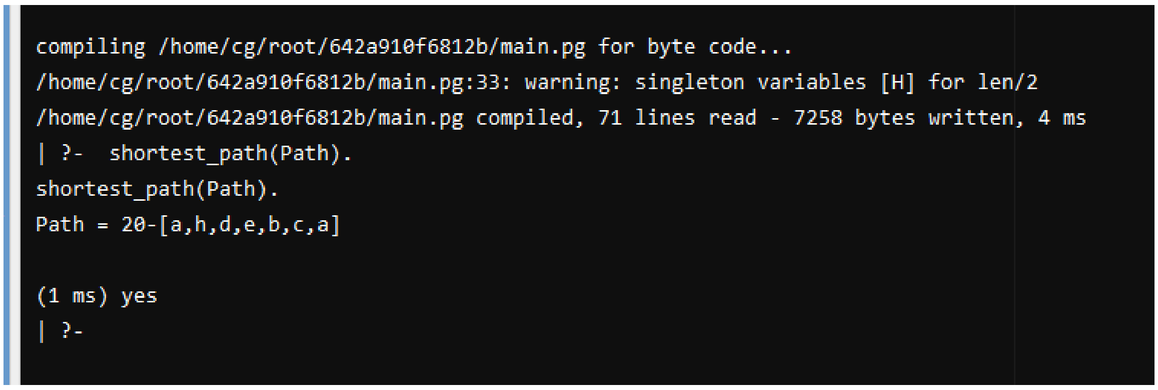
Q] Travelling Salesman Problem in Prolog.

Code:

```
/* Travelling salesman code in prolog. */
/* This is the data set.*/
  edge(a, b, 3).
  edge(a, c, 4).
  edge(a, d, 2).
  edge(a, e, 7).
  edge(b, c, 4).
  edge(b, d, 6).
  edge(b, e, 3).
  edge(c, d, 5).
  edge(c, e, 8).
  edge(d, e, 6).
  edge(b, a, 3).
  edge(c, a, 4).
  edge(d, a, 2).
  edge(e, a, 7).
  edge(c, b, 4).
  edge(d, b, 6).
  edge(e, b, 3).
  edge(d, c, 5).
  edge(e, c, 8).
  edge(e, d, 6).
  edge(a, h, 2).
  edge(h, d, 1).

  /* Finds the length of a list, while there is something in the list it increments N
      when there is nothing left it returns.*/

  len([], 0).
  len([H|T], N):- len(T, X), N is X+1 .

  /*Best path, is called by shortest_path.  It sends it the paths found in a
   path, distance format*/

  best_path(Visited, Total):- path(a, a, Visited, Total).



  /*Path is expanded to take in distance so far and the nodes visited */

  path(Start, Fin, Visited, Total) :- path(Start, Fin, [Start], Visited, 0, Total).

  /*This adds the stopping location to the visited list, adds the distance and then calls recursive
```

```
    to the next stopping location along the path */

path(Start, Fin, CurrentLoc, Visited, Costn, Total) :-
   edge(Start, StopLoc, Distance), NewCostn is Costn + Distance, \+ member(StopLoc,
CurrentLoc),
   path(StopLoc, Fin, [StopLoc|CurrentLoc], Visited, NewCostn, Total).

/*When we find a path back to the starting point, make that the total distance and make
    sure the graph has touch every node*/

path(Start, Fin, CurrentLoc, Visited, Costn, Total) :-
   edge(Start, Fin, Distance), reverse([Fin|CurrentLoc], Visited), len(Visited, Q),
   (Q\=7 -> Total is 100000; Total is Costn + Distance).

/*This is called to find the shortest path, takes all the paths, collects them in holder.
    Then calls pick on that holder which picks the shortest path and returns it*/

shortest_path(Path):-setof(Cost-Path, best_path(Path,Cost), Holder),pick(Holder,Path).

/* Is called, compares 2 distances. If cost is smaller than bcost, no need to go on. Cut it.*/

best(Cost-Holder,Bcost-_,Cost-Holder):- Cost<Bcost,!.
best(_,X,X).

/*Takes the top path and distance off of the holder and recursively calls it.*/

pick([Cost-Holder|R],X):- pick(R,Bcost-Bholder),best(Cost-Holder,Bcost-Bholder,X),!.
pick([X],X).
```

Output:



```
compiling /home/cg/root/642a910f6812b/main.pg for byte code...
/home/cg/root/642a910f6812b/main.pg:33: warning: singleton variables [H] for len/2
/home/cg/root/642a910f6812b/main.pg compiled, 71 lines read - 7258 bytes written, 4 ms
| ?-  shortest_path(Path).
shortest_path(Path).
Path = 20-[a,h,d,e,b,c,a]

(1 ms) yes
| ?-
```