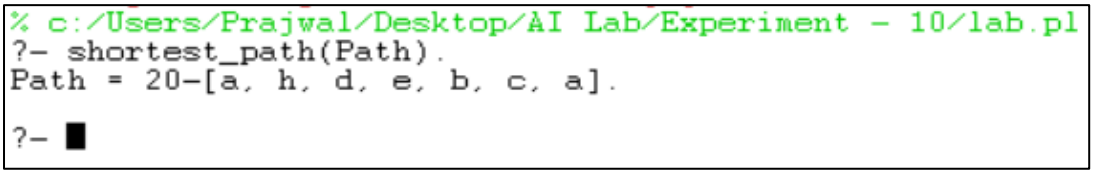
Implement traveling salesman problem using Prolog.

Output :



Find factorial of a given number using Prolog.

Output :

```
% c:/Users/Prajwal/Desktop/AI Lab/Experiment - 11/lab.pl
?- factorial(5).
120
true .

?- trace.
true.

[trace]  ?- factorial(5).
   Call: (12) factorial(5) ? creep
   Call: (13) factorial(5, _25794) ? creep
   Call: (14) 5>0 ? creep
   Exit: (14) 5>0 ? creep
   Call: (14) _28232 is 5+ -1 ? creep
   Exit: (14) 4 is 5+ -1 ? creep
   Call: (14) factorial(4, _29854) ? creep
   Call: (15) 4>0 ? creep
   Exit: (15) 4>0 ? creep
   Call: (15) _32292 is 4+ -1 ? creep
   Exit: (15) 3 is 4+ -1 ? creep
   Call: (15) factorial(3, _33914) ? creep
   Call: (16) 3>0 ? creep
   Exit: (16) 3>0 ? creep
   Call: (16) _36352 is 3+ -1 ? creep
   Exit: (16) 2 is 3+ -1 ? creep
   Call: (16) factorial(2, _37974) ? creep
   Call: (17) 2>0 ? creep
   Exit: (17) 2>0 ? creep
   Call: (17) _40412 is 2+ -1 ? creep
   Exit: (17) 1 is 2+ -1 ? creep
   Call: (17) factorial(1, _42034) ? creep
   Call: (18) 1>0 ? creep
   Exit: (18) 1>0 ? creep
   Call: (18) _44472 is 1+ -1 ? creep
   Exit: (18) 0 is 1+ -1 ? creep
   Call: (18) factorial(0, _46094) ? creep
   Exit: (18) factorial(0, 1) ? creep
   Call: (18) _42034 is 1*1 ? creep
   Exit: (18) 1 is 1*1 ? creep
   Exit: (17) factorial(1, 1) ? creep
   Call: (17) _37974 is 2*1 ? creep
   Exit: (17) 2 is 2*1 ? creep
   Exit: (16) factorial(2, 2) ? creep
   Call: (16) _33914 is 3*2 ? creep
   Exit: (16) 6 is 3*2 ? creep
   Exit: (15) factorial(3, 6) ? creep
   Call: (15) _29854 is 4*6 ? creep
   Exit: (15) 24 is 4*6 ? creep
   Exit: (14) factorial(4, 24) ? creep
   Call: (14) _25794 is 5*24 ? creep
   Exit: (14) 120 is 5*24 ? creep
   Exit: (13) factorial(5, 120) ? creep
   Call: (13) write(120) ? creep
120
   Exit: (13) write(120) ? creep
   Exit: (12) factorial(5) ? creep
true .

[trace]  ?-
|    notrace.
true.

[debug]  ?- nodebug.
true.
```