Name:Akanksha Dewangan
Roll No: MT19049

Artificial intelligence

Assignment no. 2

Note:
1.road_data.csv file contains 3 columns , as xlsx file where it contains source city, destination city and cost between them and it's like a fact for our program. (Csv made using python).
2.heuritsic is found by using the dijkstra algorithm as instructed we can use any algorithm for finding heuristic between the city and form every pair.
3. Three prolog files are uploaded where overall.pl will run both and best_first.pl and depth_first.pl separately along with this road_data.csv also their.
4.Hence there will be running code.

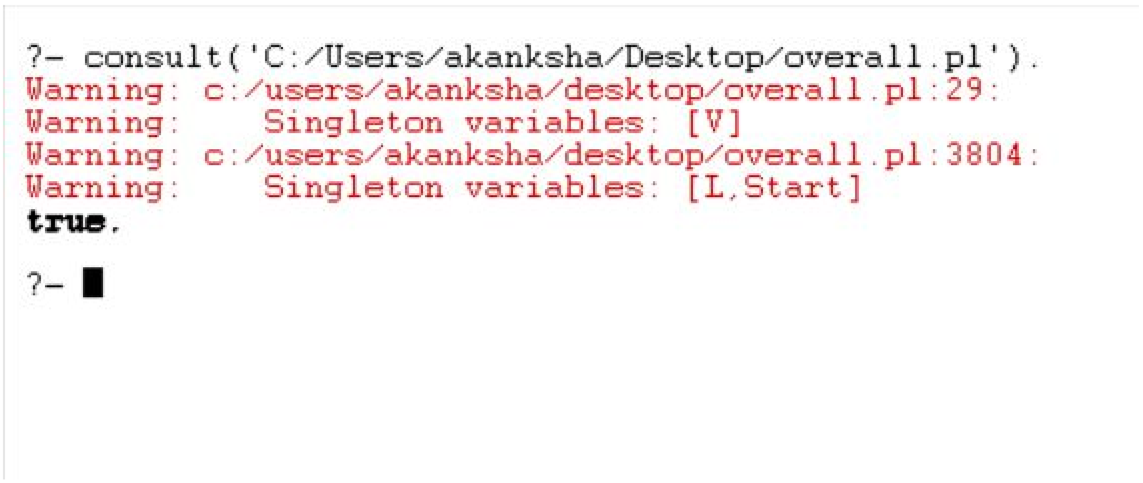
Feature used of prolog:-
I.list,
ex-maplist(assert, Rows)
ii.recursive call
iii.backtrack
Iv.arithmetic operator :- +, =< , is etc.
(above 4 points you can see in below code)
dfs_recursive(Start, Goal, Path, Visit, C):-cost(Start , Next , Value), not(member(Next, Path)), not(member(Next, Visit)),
append(Path,[Next],R), append(Visit,[Next],T),New_value is Value+C ,dfs_recursive(Next, Goal, R, T, New_value).

v.retract() and fail : to retract and keep information and store till inputs are coming temporary and fail is used after fail it will try again for the next option.
vi.assert(),memeber(),select()...aand many more.
Vii.fail,! (cut points), etc.

Code:
```
%%This is the first step by writing “main” we will start our program:
main:-
   write("Welcome to find shortest path between the distance of 2 cities or more"),nl,nl,
   write("We have 2 options of algorithm "),nl,nl,
   write("A. Depth First Serach "),nl,nl,
   write("B. Best First Search "),nl,nl,
   write("if you want to try Depth first search start writting as follow line "),nl,nl,
   write("?-go(source_city, destination_city)    "),nl,nl,
   write("else if you want to try Best first search start writting as followed sytanx "),nl,nl,
   write("?-checkCsv "),nl,nl,
   write("|: source_city."),nl,nl,
   write("|: destination_city."),nl,nl.
```

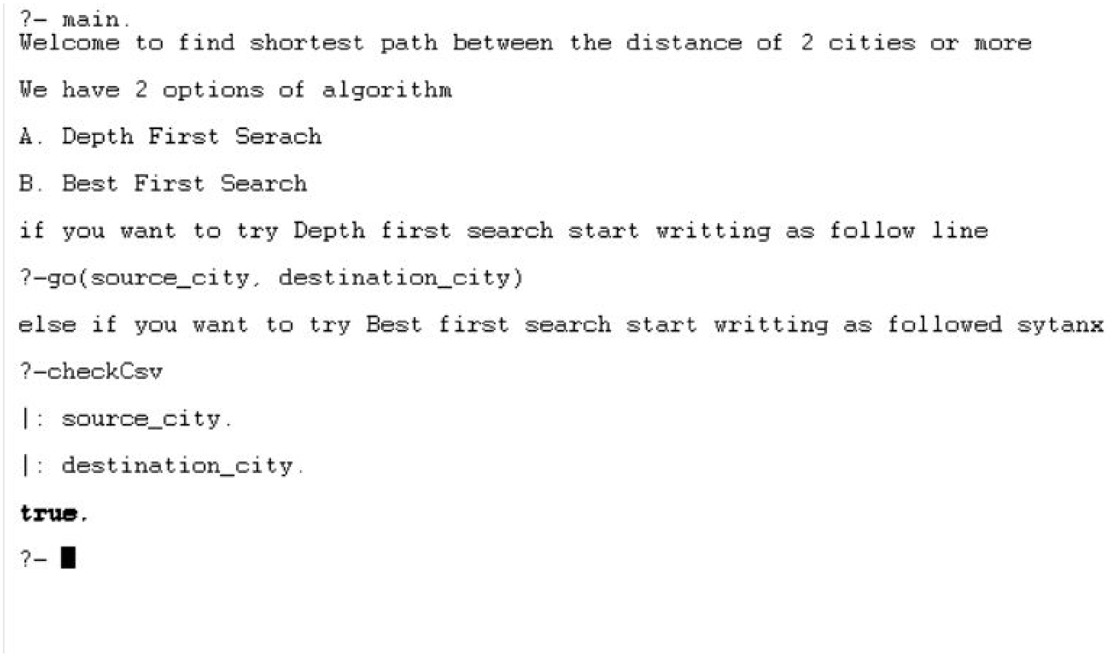
%% and as per given syntax user will start writing syntax for best first sriracha nd depth first
%% serach

Output:-



```
?- consult('C:/Users/akanksha/Desktop/overall.pl').
Warning: c:/users/akanksha/desktop/overall.pl:29:
Warning:    Singleton variables: [V]
Warning: c:/users/akanksha/desktop/overall.pl:3804:
Warning:    Singleton variables: [L,Start]
true.

?- 
```

After writing “main” program getting started:



```
?- main.
Welcome to find shortest path between the distance of 2 cities or more

We have 2 options of algorithm

A. Depth First Serach

B. Best First Search

if you want to try Depth first search start writting as follow line

?-go(source_city, destination_city)

else if you want to try Best first search start writting as followed sytanx

?-checkCsv

|: source_city.

|: destination_city.

true.

?- 
```

A.Depth first Search:-

Code:-

1.Read CSV file having 3 columns sorce city,destination city, their distance as per present in the given road distance data set.
Here each row as required is read as a fact name is “cost” and it is having 3 parameters as source city, destination city and cost.Mapping it row wise.

checkCsv:-csv_read_file('C:/Users/akanksha/Desktop/road_data.csv', Rows,[functor(cost), arity(3)]),maplist(assert, Rows),initialise.

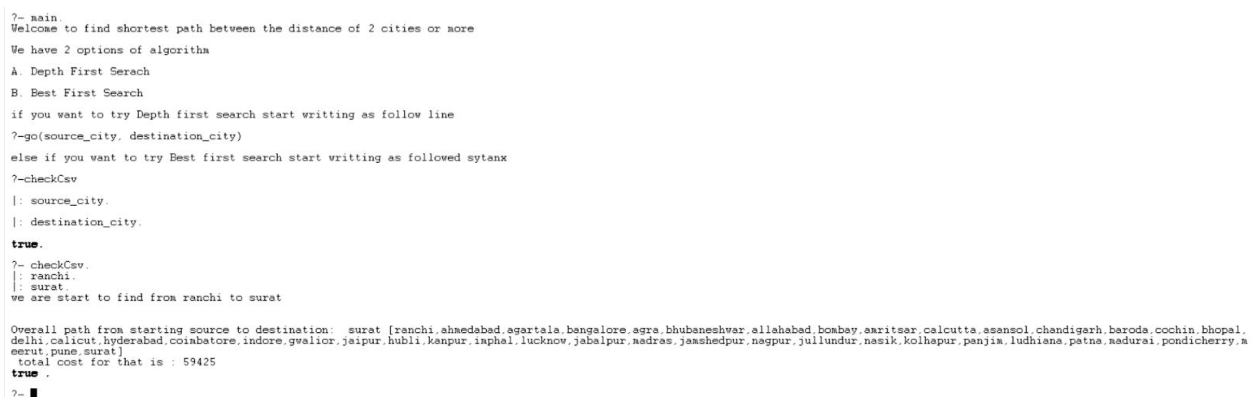
2.Base case if we reach the goal node from the start node. then it will print the output as path from source to destination and total cost for the same.
dfs_recursive(Goal, Goal, Path, Visit, C):-nl, write("Overall path from starting source to destination: "),write(" "),write(Goal),write(" "),write(Path),nl,
                              write(" total cost for that is : "),write(C).

3.This will recursively call for backtrack so that once it will not found
%% the goal city it will look for the next path after backtrack.
dfs_recursive(Start, Goal, Path, Visit, C):-cost(Start , Next , Value), not(member(Next, Path)), not(member(Next, Visit)),
append(Path,[Next],R), append(Visit,[Next],T),New_value is Value+C ,dfs_recursive(Next, Goal, R, T, New_value).

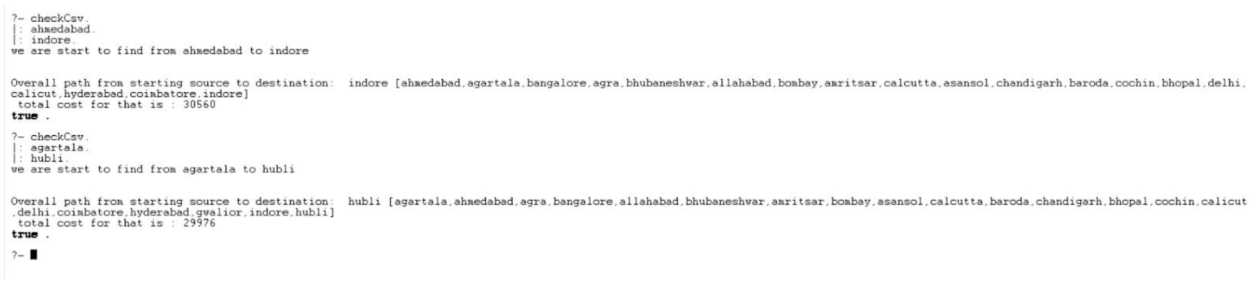
4.here it initialises which started from checkCsv call.
Initialise:-
read(Start), read(Goal), assert(goalto(Goal)),write("we are start to find from "),write(Start),write(" to "),write(Goal),nl,nl,dfs_recursive(Start, Goal, [Start],[Start], 0).

Output:-
You have to write “checkCsv” to start the program in prolog.
.After writing that you have to write source city and destination city so that we can calculate the total cost to reach each path by adding on path distance between the each state in path.

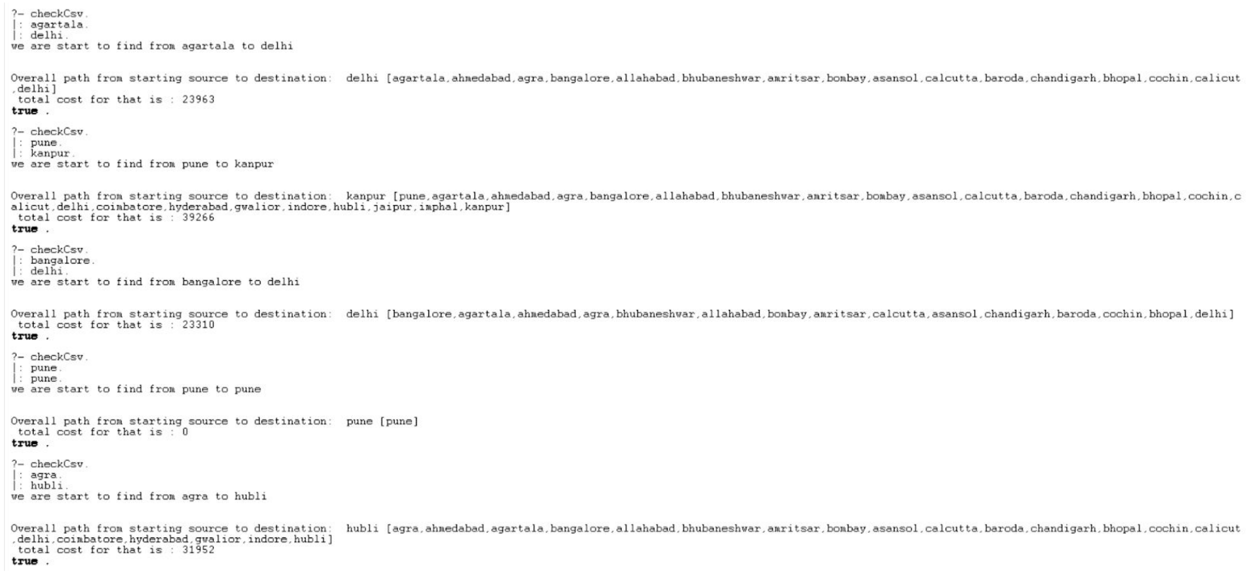
In following I cover all cases of to find distance
I.direct connected cities path and their cost.
II. indirect connection cities path and their cost.



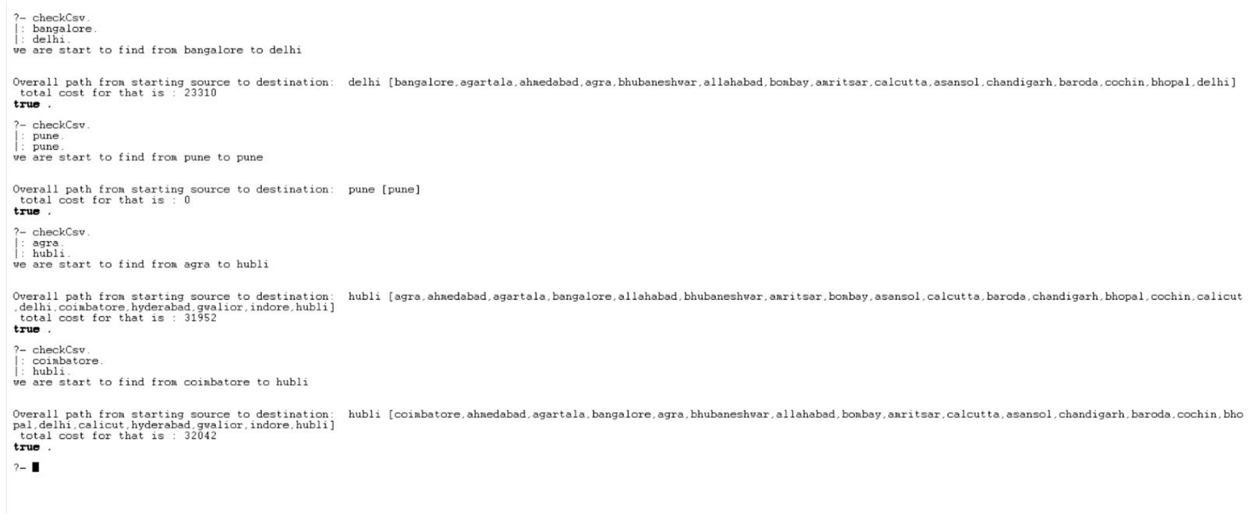
```
?- main.
Welcome to find shortest path between the distance of 2 cities or more

We have 2 options of algorithm

A. Depth First Serach

B. Best First Search

if you want to try Depth first search start writting as follow line

?-go(source_city, destination_city)

else if you want to try Best first search start writting as followed sytanx

?-checkCsv

|: source_city.

|: destination_city.

true.

?- checkCsv.
|: ranchi.
|: surat.
we are start to find from ranchi to surat


Overall path from starting source to destination:  surat [ranchi,ahmedabad,agartala,bangalore,agra,bhubaneshwar,allahabad,bombay,amritsar,calcutta,asansol,chandigarh,baroda,cochin,bhopal,
delhi,calicut,hyderabad,coimbatore,indore,gwalior,jaipur,hubli,kanpur,imphal,lucknow,jabalpur,madras,jamshedpur,nagpur,jullundur,nasik,kolhapur,panjim,ludhiana,patna,madurai,pondicherry,m
eerut,pune,surat]
 total cost for that is : 59425
true .

?- █
```



```
?- checkCsv.
|: ahmedabad.
|: indore.
we are start to find from ahmedabad to indore


Overall path from starting source to destination:  indore [ahmedabad,agartala,bangalore,agra,bhubaneshwar,allahabad,bombay,amritsar,calcutta,asansol,chandigarh,baroda,cochin,bhopal,delhi,
calicut,hyderabad,coimbatore,indore]
 total cost for that is : 30560
true .

?- checkCsv.
|: agartala.
|: hubli.
we are start to find from agartala to hubli


Overall path from starting source to destination:  hubli [agartala,ahmedabad,agra,bangalore,allahabad,bhubaneshwar,amritsar,bombay,asansol,calcutta,baroda,chandigarh,bhopal,cochin,calicut
,delhi,coimbatore,hyderabad,gwalior,indore,hubli]
 total cost for that is : 29976
true .

?- █
```



```
?- checkCsv.
|: agartala.
|: delhi.
we are start to find from agartala to delhi


Overall path from starting source to destination:  delhi [agartala,ahmedabad,agra,bangalore,allahabad,bhubaneshwar,amritsar,bombay,asansol,calcutta,baroda,chandigarh,bhopal,cochin,calicut
,delhi]
 total cost for that is : 23963
true .

?- checkCsv.
|: pune.
|: kanpur.
we are start to find from pune to kanpur


Overall path from starting source to destination:  kanpur [pune,agartala,ahmedabad,agra,bangalore,allahabad,bhubaneshwar,amritsar,bombay,asansol,calcutta,baroda,chandigarh,bhopal,cochin,c
alicut,delhi,coimbatore,hyderabad,gwalior,indore,hubli,jaipur,imphal,kanpur]
 total cost for that is : 39266
true .

?- checkCsv.
|: bangalore.
|: delhi.
we are start to find from bangalore to delhi


Overall path from starting source to destination:  delhi [bangalore,agartala,ahmedabad,agra,bhubaneshwar,allahabad,bombay,amritsar,calcutta,asansol,chandigarh,baroda,cochin,bhopal,delhi]
 total cost for that is : 23310
true .

?- checkCsv.
|: pune.
|: pune.
we are start to find from pune to pune


Overall path from starting source to destination:  pune [pune]
 total cost for that is : 0
true .

?- checkCsv.
|: agra.
|: hubli.
we are start to find from agra to hubli


Overall path from starting source to destination:  hubli [agra,ahmedabad,agartala,bangalore,allahabad,bhubaneshwar,amritsar,bombay,asansol,calcutta,baroda,chandigarh,bhopal,cochin,calicut
,delhi,coimbatore,hyderabad,gwalior,indore,hubli]
 total cost for that is : 31952
true .
```



```
?- checkCsv.
|: bangalore.
|: delhi.
we are start to find from bangalore to delhi


Overall path from starting source to destination:  delhi [bangalore,agartala,ahmedabad,agra,bhubaneshwar,allahabad,bombay,amritsar,calcutta,asansol,chandigarh,baroda,cochin,bhopal,delhi]
 total cost for that is : 23310
true .

?- checkCsv.
|: pune.
|: pune.
we are start to find from pune to pune


Overall path from starting source to destination:  pune [pune]
 total cost for that is : 0
true .

?- checkCsv.
|: agra.
|: hubli.
we are start to find from agra to hubli


Overall path from starting source to destination:  hubli [agra,ahmedabad,agartala,bangalore,allahabad,bhubaneshwar,amritsar,bombay,asansol,calcutta,baroda,chandigarh,bhopal,cochin,calicut
,delhi,coimbatore,hyderabad,gwalior,indore,hubli]
 total cost for that is : 31952
true .

?- checkCsv.
|: coimbatore.
|: hubli.
we are start to find from coimbatore to hubli


Overall path from starting source to destination:  hubli [coimbatore,ahmedabad,agartala,bangalore,agra,bhubaneshwar,allahabad,bombay,amritsar,calcutta,asansol,chandigarh,baroda,cochin,bho
pal,delhi,calicut,hyderabad,gwalior,indore,hubli]
 total cost for that is : 32042
true .

?- █
```

B.Best first Search
Here heuristic is taken out using Dijkstra's single source shortest path.

Code:

%% here a table is created which will keep the information of visited or unvisited of the current city while
%%finding path from source to destination city.
%% if city iis visited then we will update it with 1 from 0, else if it is unvisited then we will left it 0 as it was initilaised.

%% here assert is used to add into the database which is table_cityname_visit here.
%% also their is a distance list[] is asserted which will keep infoemation of distance of visited city which
%% are currently in visited mode.

table_cityname_visit :-

assert(visit(jaipur,0)),assert(visit(pune,0)),assert(visit(gwalior,0)),assert(visit(vijayawada,0)),asse
rt(visit(kolhapur,0)),assert(visit(indore,0)),

assert(visit(agartala,0)),assert(visit(hyderabad,0)),assert(visit(delhi,0)),assert(visit(ahmedabad,0
)),assert(visit(cochin,0)),assert(visit(surat,0)),

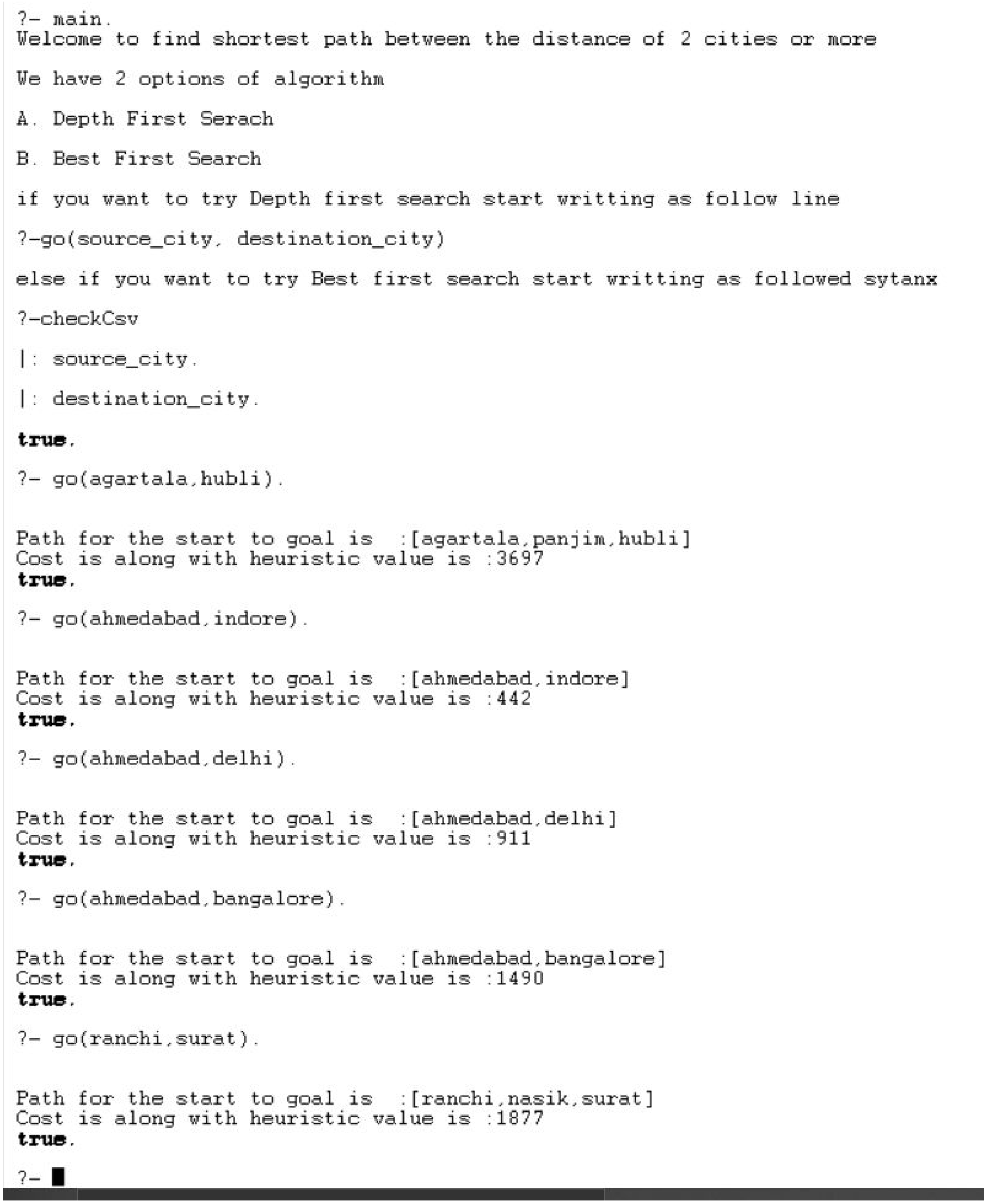
assert(visit(jamshedpur,0)),assert(visit(ludhiana,0)),assert(visit(imphal,0)),assert(visit(bhopal,0)),
assert(visit(calcutta,0)),assert(visit(allahabad,0)),

```
assert(visit(patna,0)),assert(visit(chandigarh,0)),assert(visit(bangalore,0)),assert(visit(agra,0)),as
sert(visit(bombay,0)),assert(visit(shimla,0)),

assert(visit(nagpur,0)),assert(visit(baroda,0)),assert(visit(asansol,0)),assert(visit(nasik,0)),assert
(visit(hubli,0)),

assert(visit(panjim,0)),assert(visit(vishakapatnam,0)),assert(visit(trivandrum,0)),assert(visit(kanp
ur,0)),assert(visit(shillong,0)),

assert(visit(jabalpur,0)),assert(visit(madras,0)),assert(visit(coimbatore,0)),assert(visit(madurai,0)
),assert(visit(jullundur,0)),

assert(visit(calicut,0)),assert(visit(ranchi,0)),assert(visit(meerut,0)),assert(visit(varanasi,0)),asser
t(visit(pondicherry,0)),

assert(visit(lucknow,0)),assert(visit(amritsar,0)),assert(visit(bhubaneshwar,0)),assert(visited_city
_dist([])).


%% here is the working of best_first_search started where we intially take input of start and goal
city.
%% there is if condition if start is equal to goal then we will call res(L) which will print the path is
itself a start node.
%% else we will look for recursion in best first search call;
%% steps are:
%% 1. we will look up the table cityname visited and retract it details
%% 2. and assert its value from o to 1 as now start node is now in visiting mode.
%% 3. here after updation in visited database we will have to enqueue or assert here the details
in queue which is list
%% here containing details as [cur_node,initialstart_infinite vasle as 900000 here,[cur_node]
insert in list, cost initialise to 0]
%% 4.after this recursion is call for best_first_search()
go(Start,Goal):-
            retractall(visit(_,_)),
            retractall(visited_city_dist(_)),
            retractall(queue(_,_,_,_)),
           %%there is if else condition here
            (Start=Goal -> res(Start) ;

table_cityname_visit,retract(visit(Start,0)),assert(visit(Start,1)),assert(queue(Start,900000,[Start],
0)),best_first_search(Start,Goal)),!.
```

```
%% 5. here best first search calling is started in recursion
%% 6.it will first look for all the adjacent children nodes which are present in the graph. It will
call childrens_of_current(Start,Goal).
%% 7.step 7 called childrens_of_current(Start,Goal).

%%  7.here childrens of current node are getting store
%% we will add the cost of start and next city distance and mark them as visited and then
%% look for heuristic add their values and add them in queue as a all childrens and in queue
cost is the new added cost
%% now from here we got the adjancent childrens of node.
childrens_of_current(Start,Goal) :-
                    city_connection_cost(Start,Next,C),
                    h(Next,Goal,Hcost),
                    (visit(Next,0) ->
                        retract(visit(Next,_)),
                        assert(visit(Next,1)),
                        queue(Start,_,List,Cost),
                        append(List,[Next],New_List),
                        New_cost is C+Cost,
                        assert(queue(Next,Hcost,New_List,New_cost))),
                    retract(visited_city_dist(X)),
                    append(X,[Hcost],New_X),
                    assert(visited_city_dist(New_X)),
                    fail.


%%this is result printing clause for initial base case.
res(L):- write("Path  is"),write(Start).








%% 8.step is visited city distance database is retract and
%% 9. after that it will take out the minimum cost value city from the list of
%% current children nodes and add into the next visiting city.
%% 10.here there will be a if condition if we reach the goal then print path and cost else we will
call the best first search
%% recursively.
```

```
print_res(List,C):- nl,nl,write("Path for the start to goal is  :"),write(List),nl,write("Cost is along
with heuristic value is :"),write(C).

best_first_search(Start,Goal) :-
childrens_of_current(Start,Goal);visited_city_dist(Y),(min_list(Y,Min) ->
select(Min,Y,Updated_Y),retract(visited_city_dist(_)),
                                        assert(visited_city_dist(Updated_Y)),
                                        retract(queue(Next,Min,List,C)),
                                        assert(queue(Next,900000,List,C)),
                              %% if condition if we reach goal city then print the output and call
print_res
                              %% else we will call for best first search function recursivly.
                                        (  Next=Goal-> print_res(List,C)
;best_first_search(Next,Goal))).
```

Output:-



```
?- main.
Welcome to find shortest path between the distance of 2 cities or more

We have 2 options of algorithm

A. Depth First Serach

B. Best First Search

if you want to try Depth first search start writting as follow line

?-go(source_city, destination_city)

else if you want to try Best first search start writting as followed sytanx

?-checkCsv

|: source_city.

|: destination_city.

true.

?- go(agartala,hubli).


Path for the start to goal is  :[agartala,panjim,hubli]
Cost is along with heuristic value is :3697
true.

?- go(ahmedabad,indore).


Path for the start to goal is  :[ahmedabad,indore]
Cost is along with heuristic value is :442
true.

?- go(ahmedabad,delhi).


Path for the start to goal is  :[ahmedabad,delhi]
Cost is along with heuristic value is :911
true.

?- go(ahmedabad,bangalore).


Path for the start to goal is  :[ahmedabad,bangalore]
Cost is along with heuristic value is :1490
true.

?- go(ranchi,surat).


Path for the start to goal is  :[ranchi,nasik,surat]
Cost is along with heuristic value is :1877
true.

?- █
```

References:

1.https://stackoverflow.com/questions/26082499/how-can-i-use-assert-in-prolog
2.https://stackoverflow.com/questions/22591030/prolog-read-a-csv-file-and-make-a-predicate-findall
3.https://www.cs.unm.edu/~luger/ai-final/code/PROLOG.best.html
4.https://www.cs.unm.edu/~luger/ai-final/code/PROLOG.depth.html
5.https://swish.swi-prolog.org/p/Graphs1.swinb
6. https://swish.swi-prolog.org/p/Tree%20search.swinb
7.https://stackoverflow.com/questions/27065774/depth-first-search-algorithm-prolog
8.https://github.com/indrefi/Prolog-Personal-Work/blob/master/Graph-Search-Algorithms.pl
9.https://www.goeduhub.com/7451/write-prolog-program-first-search-applied-eight-puzzle-problem
10. http://www.cs.ukzn.ac.za/~hughm/ai/notes/prologsearch.pdf
11.https://www.csee.umbc.edu/courses/771/current/presentations/prolog%20search.pdf
12.ttp://www.cs.ubbcluj.ro/~csatol/log_funk/prolog/slides/7-search.pdf
13.https://medium.com/@dpthegrey/best-first-search-heuristic-search-technique-717f0b761559
14.http://users.utcluj.ro/~cameliav/lp/11_Graphs_Search.pdf
15.https://www.cs.unm.edu/~luger/ai-final2/CH4_Depth-.%20Breadth-,%20and%20Best-first%20Search.pdf
16.http://www.cs.ukzn.ac.za/~hughm/ai/notes/prologsearch.pdf
17.http://users.utcluj.ro/~cameliav/lp/11_Graphs_Search.pdf
18.https://github.com/bonhamdaniel/prolog-basics/blob/master/minimum-path/minimumpath.pl
19.any many more...