**Questions:**

1. **Define a recursive procedure in Python and in Prolog to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.**
2. **Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.**
3. **Modify the Python and Prolog codes demonstrated above to find h2 and h3 discussed above.**

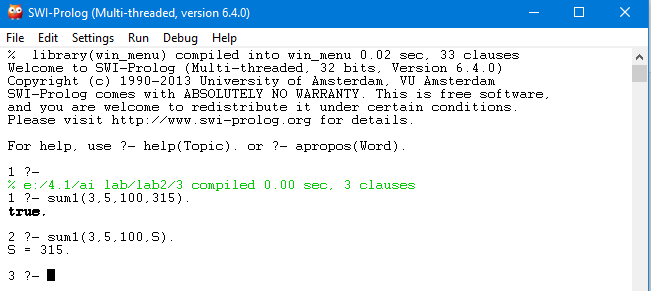
**Solution to the question no 1**

The demonstrated Prolog code to find the sum is as below:

sum1(1,\_,F,F):-!.

sum1(N,I,F,S):-N>0, N1 is N-1, sum1(N1,I,F,S1) , S is S1+F+N1\*I.

A sample of input and output is as below:



The demonstrated Python code to find sum is as below:

def sum(n,i,f):

if(n == 0):

return 0

elif(n >= 1):

return sum(n-1,i,f)+f+(n-1)\*i

#main

fterm = int(input('First Term:'))

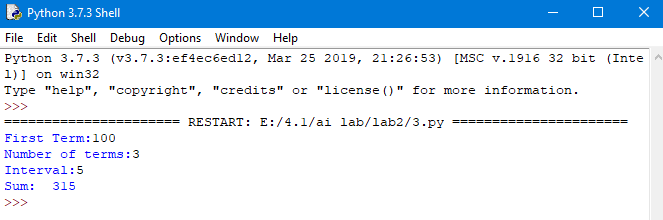
numterm = int(input('Number of terms:'))

inte = int(input('Interval:'))

total = sum(numterm,fterm,inte)

print('Sum: ', total)

A sample of input and output is as below:



**Solution to the question no 2**

The demonstrated Prolog code is below:

neighbor(i,a,35). neighbor(i,b,45). neighbor(a,c,22).

neighbor(a,d,32). neighbor(b,d,28). neighbor(b,e,36).

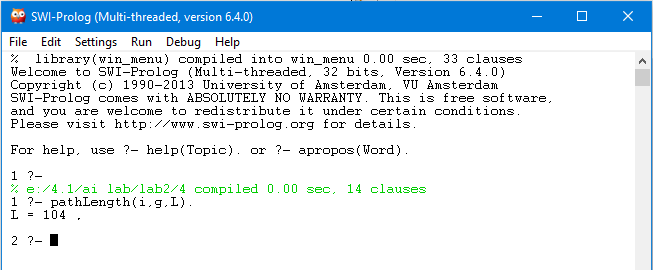
neighbor(b,f,27). neighbor(c,d,31). neighbor(c,g,47).

neighbor(d,g,30). neighbor(e,g,26).

pathLength(X,Y,L):- neighbor(X,Y,L),!.

pathLength(X,Y,L):- neighbor(X,Z,L1), pathLength(Z,Y,L2), L is L1+L2.

A sample of input and output is as below:



The demonstrated Python code is below:

from collections import defaultdict

def dfs(source,dest,visited,path):

visited[source]= True

path.append(source)

if source == dest:

total =0

print(path)

l = len(path)

for i in range(l-1):

total += graph[path[i]][path[i+1]]

print(total)

else:

for i in graph[source]:

if visited[i] == False:

dfs(i,dest,visited,path)

path.pop()

visited[source]=False

graph = defaultdict(dict)

graph[0][1]=35

graph[0][2]=45

graph[1][3]=22

graph[1][4]=32

graph[2][4]=28

graph[2][5]=36

graph[2][6]=27

graph[4][7]=30

graph[3][4]=31

graph[3][7]=47

graph[4][7]=30

graph[5][7]=26

source = int(input("Source: "))

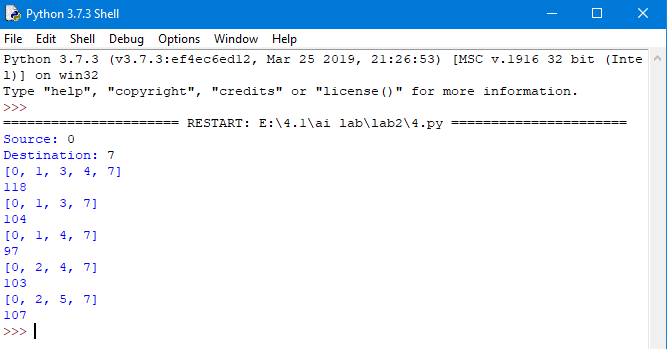
dest = int(input("Destination: "))

visited = [False]\*8

path=[]

dfs(source,dest, visited, path)

A sample of input and output is as below:



**Solution to the question no 3**

The demonstrated Prolog code to find heuristic function(h1) of 8-puzzle problem is as below:

gtp(1,1,1). gtp(2,1,2). gtp(3,1,3). gtp(4,2,3). gtp(5,3,3). gtp(6,3,2). gtp(7,3,1). gtp(8,2,1). gblnk(2,2).

tp(1,1,2). tp(2,1,3). tp(3,2,1). tp(4,2,3). tp(5,3,3). tp(6,2,2). tp(7,3,2). tp(8,1,1). blnk(3,1).

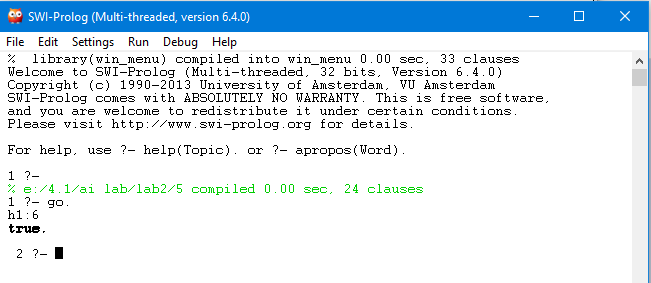
go:- catcH(1,0,H), write('h1:'),write(H).

catcH(9,X,X):-!.

catcH(T,X,Y):- check(T,V), X1 is X+V, T1 is T+1, catcH(T1,X1,Y).

check(T,V):- tp(T,A,B), gtp(T,C,D), A=C, B=D, V is 0, !.

check(\_,1):-!.

A sample of input and output is as below:

The demonstrated Python code to find heuristic function(h1) of 8-puzzle problem is as below:

gtp=[(1,1,1), (2,1,2), (3,1,3), (4,2,3), (5,3,3), (6,3,2), (7,3,1), (8,2,1)]

gblnk = (2,2)

tp=[(1,1,2), (2,1,3), (3,2,1), (4,2,3), (5,3,3), (6,2,2), (7,3,2), (8,1,1)]

blnk = (3,1)

# Procedure to find the number of mismatches

i,h=0,0

while(i<=7):

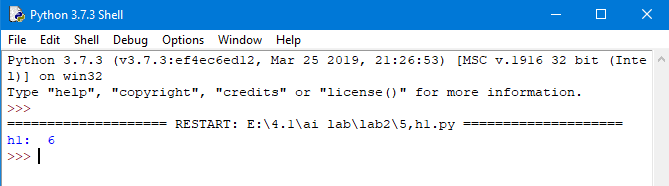
if (gtp[i][1] != tp[i][1]) | (gtp[i][2] != tp[i][2]):

h+=1

i+=1

print("h1: ", h)

A sample of input and output is as below:



The demonstrated Prolog code to find heuristic function(h2) of 8-puzzle problem is as below:

gtp(1,1,1). gtp(2,1,2). gtp(3,1,3). gtp(4,2,3). gtp(5,3,3). gtp(6,3,2). gtp(7,3,1). gtp(8,2,1). gblnk(2,2).

tp(1,1,2). tp(2,1,3). tp(3,2,1). tp(4,2,3). tp(5,3,3). tp(6,2,2). tp(7,3,2). tp(8,1,1). blnk(3,1).

go:- catcH(1,0,H), write('h1:'),write(H).

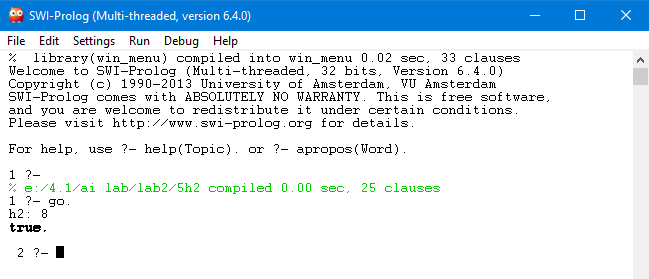
catcH(9,X,X):-!.

catcH(T,X,Y):- check(T,V), X1 is X+V, T1 is T+1, catcH(T1,X1,Y).

check(T,V):- tp(T,A,B), gtp(T,C,D), A=C, B=D, V is 0, !.

check(\_,1):-!.

A sample of input and output is as below:



The demonstrated Python code to find heuristic function(h2) of 8-puzzle problem is as below:

gtp=[(1,1,1), (2,1,2), (3,1,3), (4,2,3), (5,3,3), (6,3,2), (7,3,1), (8,2,1)]

gblnk = (2,2)

tp=[(1,1,2), (2,1,3), (3,2,1), (4,2,3), (5,3,3), (6,2,2), (7,3,2), (8,1,1)]

blnk = (3,1)

# Procedure to find the number of movements

i,h=0,0

while(i<=7):

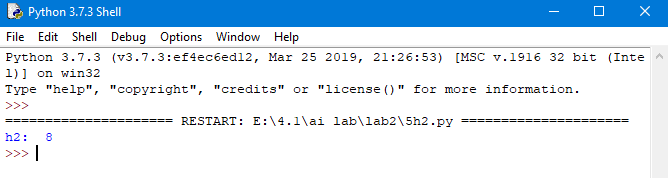
if ((gtp[i][1] != tp[i][1])|(gtp[i][2] != tp[i][2])):

h += abs(gtp[i][1] - tp[i][1]) + abs(gtp[i][2] - tp[i][2])

i += 1

print('h2: ',h)

A sample of input and output is as below:



The demonstrated Prolog code to find heuristic function(h3) of 8-queen problem is as below:

:-dynamic(hval/1).

/\* Evaluates a 8-queens' state given as list of 8 digits \*/

evalState(L,V):- assert(hval(0)),hl(1,L), di\_up(1,L),di\_dn(1,L),hval(V),

retractall(hval(\_)).

hl(8,\_):-!. hl(I,L):- nthel(I,L,X), chk\_incr(I,L,X), I1 is I+1, hl(I1,L).

chk\_incr(8,\_,\_):-!. chk\_incr(I,L,X):- I1 is I+1, nthel(I1,L,Y),

do\_incr(X,Y),chk\_incr(I1,L,X).

do\_incr(X,Y):- X=Y, incr\_hval. do\_incr(\_,\_).

incr\_hval:-hval(V), V1 is V+1, retract(hval(\_)), assert(hval(V1)).

di\_up(8,\_):-!. di\_up(I,L):- nthel(I,L,X), chkup\_incr(I,L,X,0), I1 is I+1,

di\_up(I1,L).

chkup\_incr(8,\_,\_,\_):-!.

chkup\_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1,

doup\_incr(X,Y,K1), chkup\_incr(I1,L,X,K1).

doup\_incr(X,Y,K1):- X1 is X+K1, Y=X1, incr\_hval. doup\_incr(\_,\_,\_).

di\_dn(8,\_):-!. di\_dn(I,L):- nthel(I,L,X), chkdn\_incr(I,L,X,0), I1 is I+1,

di\_dn(I1,L).

chkdn\_incr(8,\_,\_,\_):-!.

chkdn\_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1,

dodn\_incr(X,Y,K1), chkdn\_incr(I1,L,X,K1).

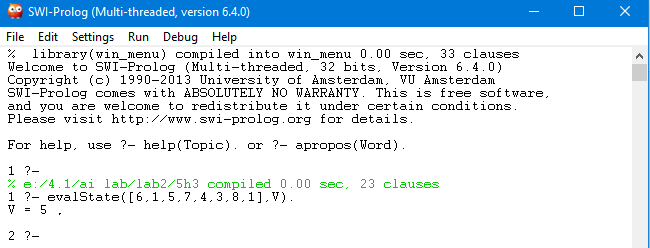
dodn\_incr(X,Y,K1):- X1 is X-K1, Y=X1, incr\_hval. dodn\_incr(\_,\_,\_).

% A procedure to find the nth element of a list

nthel(N,[\_|T],El):- N1 is N-1, nthel(N1,T,El).

nthel(1,[H|\_],H):-!.

A sample of input and output is as below:



The demonstrated Python code to find heuristic function(h3) of 8-queen problem is as below:

#procedure to find out heuristic function(h3) for 8-Queens problem

state = [6,1,5,7,4,3,8,1]

total = 0

for i in range(len(state)):

temp = state[i]

for k in range(i+1, len(state),1):

if(temp == state[k]):

total +=1

j = 1

for k in range(i+1, len(state), 1):

if((temp + j < len(state)) & (temp - j > -1)):

if((state[k] == temp + j) | ( state[k] == temp - j)):

total+=1

state[k] = 9

j+=1

m = 1

for k in range(i-1, -1, -1):

if((temp + m < len(state)) & (temp - m > -1)):

if((state[k] == temp + m) | (state[k] == temp - m)):

total+=1

state[k] = 9

m+=1

print('h3:',total)

A sample of input and output is as below:

