Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE 4108

Artificial Intelligence

Submitted By:

Anika Tanzim 16.02.04.072

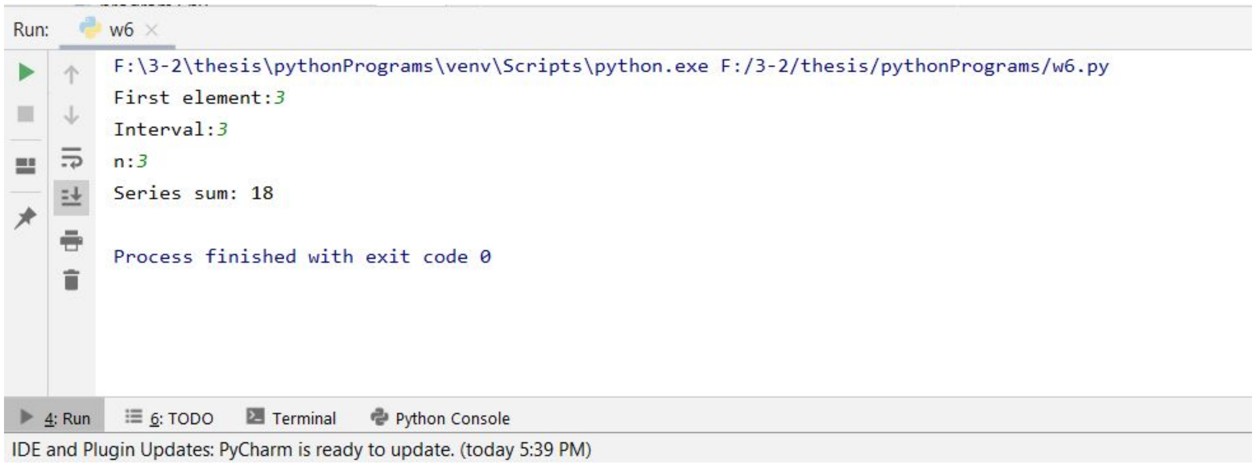
Date of Submission: **4th Match, 2020**

Q. 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.

<u>Python code:</u>

```
def recurcive_sum(N,I,F):
 if (N==0):
     return 0
 elif (N>=1):
     return recurcive_sum(N-1,I,F)+F+(N-1)*I


f=int(input('First element:'))
d=int(input('Interval:'))
n=int(input('n:'))
print('Series sum:', recurcive_sum(n,d,f))
```

<u>Output for python:</u>



```
F:\3-2\thesis\pythonPrograms\venv\Scripts\python.exe F:/3-2/thesis/pythonPrograms/w6.py
First element:3
Interval:3
n:3
Series sum: 18

Process finished with exit code 0
```

Run: w6
4: Run
6: TODO
Terminal
Python Console
IDE and Plugin Updates: PyCharm is ready to update. (today 5:39 PM)

Prolog code:

ssum(0,_,_,S):- S is 0.

ssum(N,F,I,S):- N1 is N-1,ssum(N1,F,I,S1),S is S1+F+((N-1)*I).

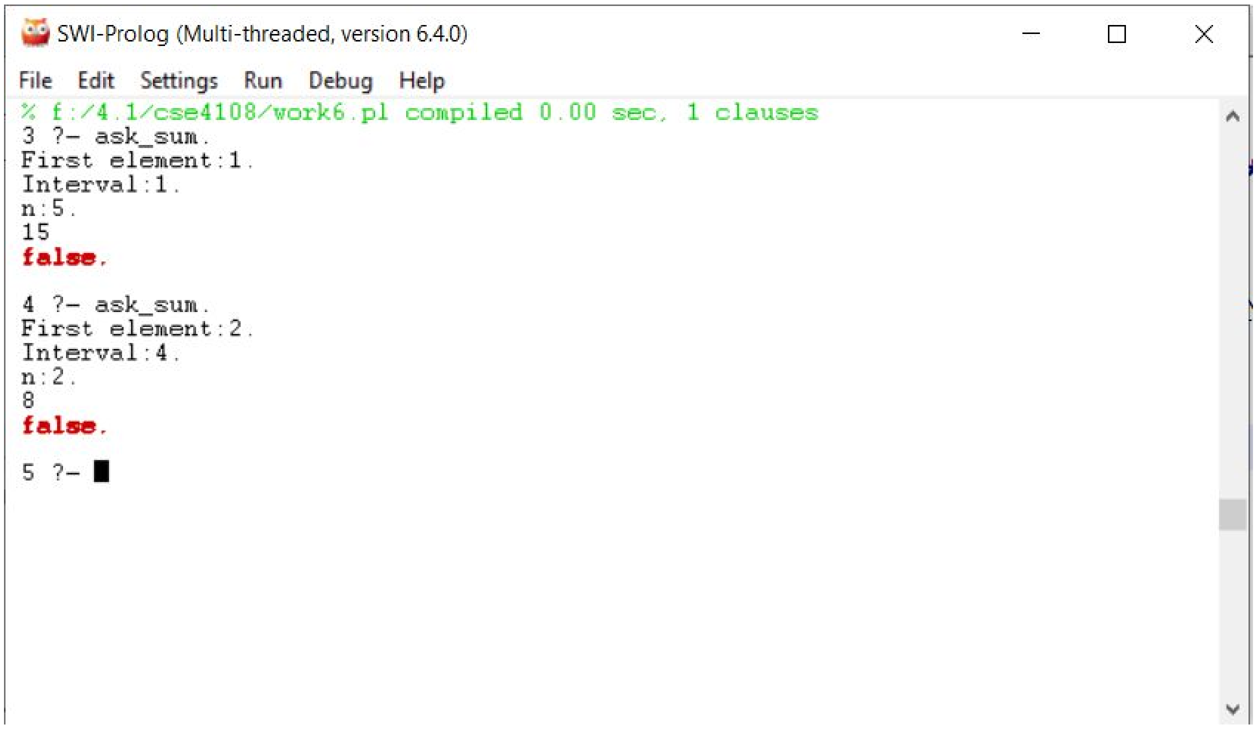
ask_sum :-

write('First element:'),read(F),

write('Interval:'),read(I),

write('n:'),read(N),ssum(N,F,I,S),write(S), nl, fail.

Output for prolog:



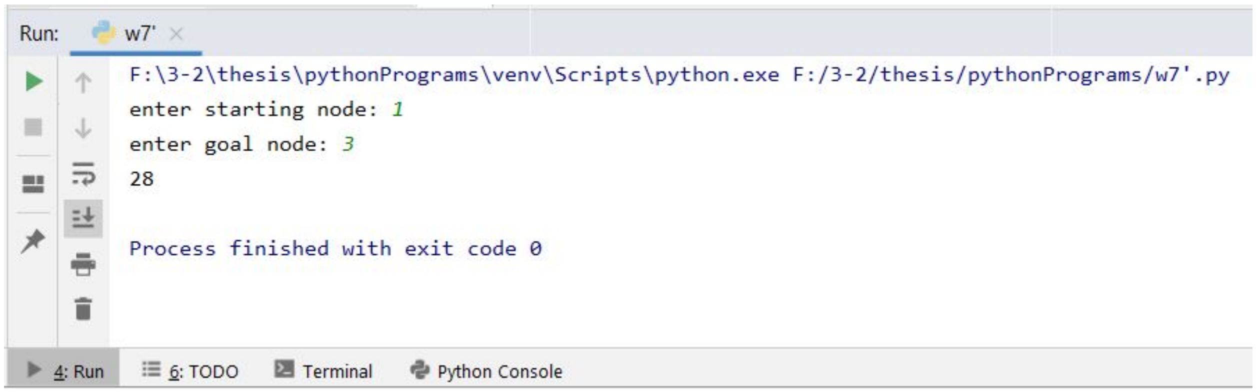
```
% f:/4.1/cse4108/work6.pl compiled 0.00 sec, 1 clauses
3 ?- ask_sum.
First element:1.
Interval:1.
n:5.
15
false.

4 ?- ask_sum.
First element:2.
Interval:4.
n:2.
8
false.

5 ?- 
```

Q. Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.

Python code:

```
graph = [(8,0,35),(8,1,45),(0,2,22),

        (0,3,32),(1,3,28),(1,4,36),

        (1,5,27),(2,3,31),(2,6,47),

        (3,6,30),(4,6,26)]


initial = int(input("enter starting node: "))

goal = int(input("enter goal node: "))

path =0

i=0

neighbour =0

x=graph[0][0]

while(i<=10):

  if(graph[i][0]==initial and graph[i][1]==goal):

      path =graph[i][2]

      break

  elif(graph[i][0]==initial):

      initial=graph[i][1]

      path = path+graph[i][2]

      #print(path)

      if(initial==goal):
        break

  i=i+1
print(path)
```

<u>Output for python code:</u>



```
F:\3-2\thesis\pythonPrograms\venv\Scripts\python.exe F:/3-2/thesis/pythonPrograms/w7'.py
enter starting node: 1
enter goal node: 3
28

Process finished with exit code 0
```

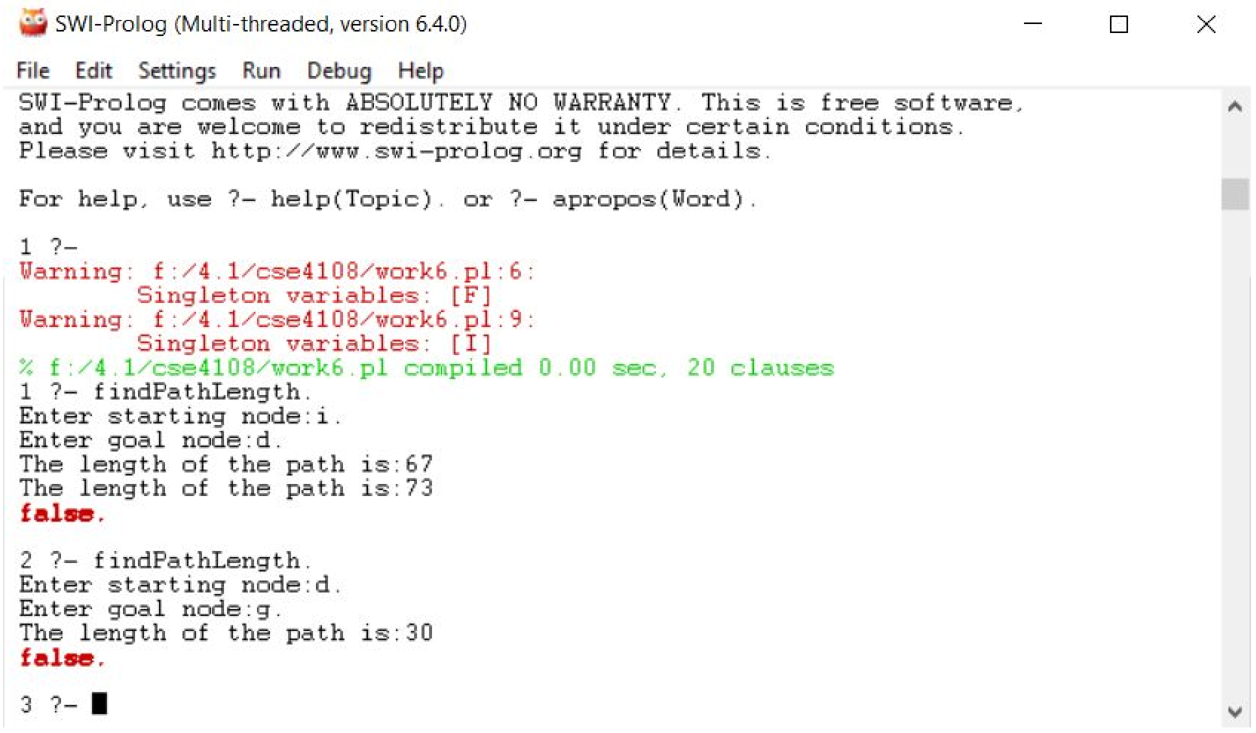
<u>Prolog code:</u>

```
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.
findPathLength:-
write('Enter starting node:'), read(X),
write('Enter goal node:'), read(Y),
```

pathLength(X,Y,L),write('The length of the path is:'),

write(L),nl,fail.

<u>Output of prolog:</u>



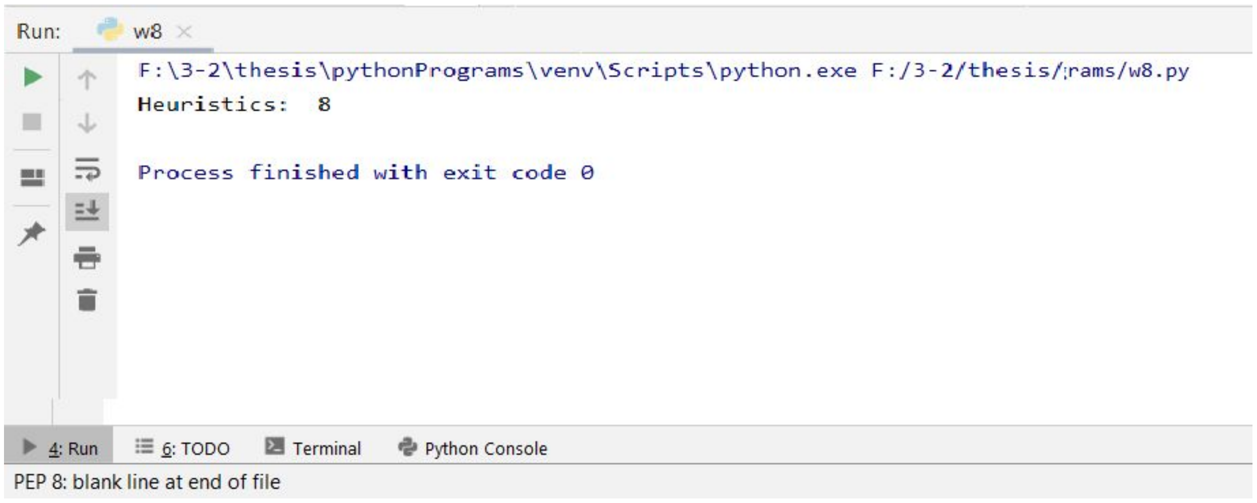
```
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?-
Warning: f:/4.1/cse4108/work6.pl:6:
        Singleton variables: [F]
Warning: f:/4.1/cse4108/work6.pl:9:
        Singleton variables: [I]
% f:/4.1/cse4108/work6.pl compiled 0.00 sec, 20 clauses
1 ?- findPathLength.
Enter starting node:i.
Enter goal node:d.
The length of the path is:67
The length of the path is:73
false.

2 ?- findPathLength.
Enter starting node:d.
Enter goal node:g.
The length of the path is:30
false.

3 ?- 
```

Q. Modify the Python and Prolog codes demonstrated above to find h_2 discussed above

<u>Python code:</u>

```
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, 1)

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)

h=0

i=0

while (i <= 7):
  if ((gtp[i][0] == tp[i][0]) and ((gtp[i][0] != tp[i][1]) or (gtp[i][2] != tp[i][2]))):
     h = h +  abs(tp[i][1] - gtp[i][1]) + abs(tp[i][2] - gtp[i][2])
  i = i + 1
print('Heuristics: ', h)
```

Output for python code:



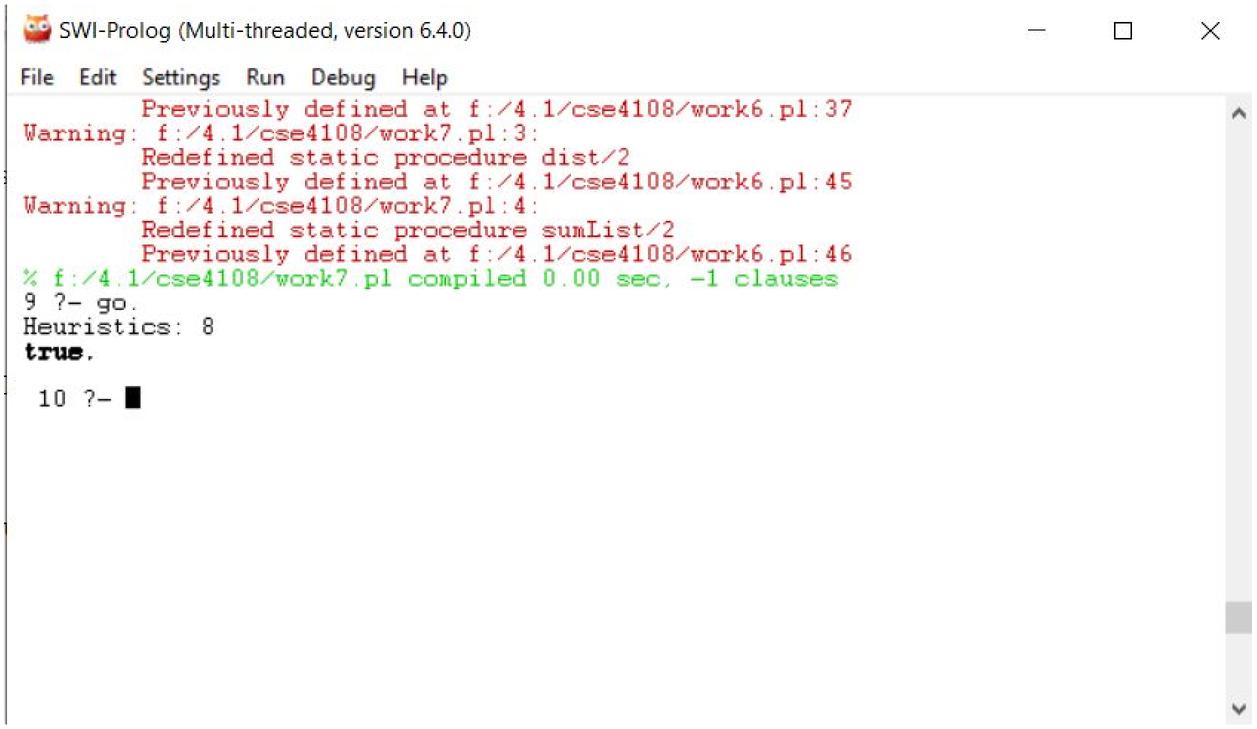
```
F:\3-2\thesis\pythonPrograms\venv\Scripts\python.exe F:/3-2/thesis/;rams/w8.py
Heuristics:   8

Process finished with exit code 0
```

PEP 8: blank line at end of file

Prolog code:

```
go:- calcH(1,[],L), sumList(L,V),write('Heuristics: '),write(V).

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

calcH(T,X,Y):- dist(T,D), append(X,[D],X1), T1 is T+1, calcH(T1,X1,Y).

dist(T,V):-tp(T,A,B), gtp(T,C,D), V is abs(A-C) + abs(B-D).

sumList([],0):-!.

sumList(L,V):-L=[H|T], sumList(T,V1), V is V1+H.
```

Output for prolog code:



```
        Previously defined at f:/4.1/cse4108/work6.pl:37
Warning: f:/4.1/cse4108/work7.pl:3:
        Redefined static procedure dist/2
        Previously defined at f:/4.1/cse4108/work6.pl:45
Warning: f:/4.1/cse4108/work7.pl:4:
        Redefined static procedure sumList/2
        Previously defined at f:/4.1/cse4108/work6.pl:46
% f:/4.1/cse4108/work7.pl compiled 0.00 sec, -1 clauses
9 ?- go.
Heuristics: 8
true.

 10 ?-
```