Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE4108: Artificial Intelligence Lab

Name: Rakib Hossain Rifat

ID: 16.02.04.099

Group: B2

Assignment # **02**

Date of Submission: 11/03/2020

QUESTION-1:

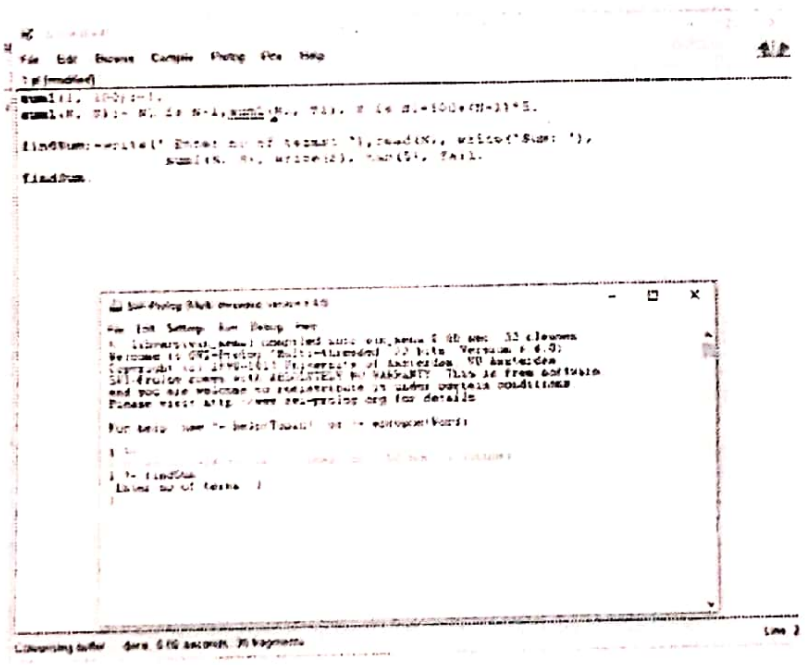
Define a recursive procedure in Python and in Prolog to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.

ANSWER:

Prolog Code:

```
sum1(1, 100):-!.

sum1(N, S):- N1 is N-1, sum1(N1, S1), S is S1+100+(N-1)*5.


findSum:-write(' Enter no of terms: '),read(N), write('Sum: '),
                sum1(N, S), write(S), tab(5), fail.

findSum.
```

Sample Input/Output:

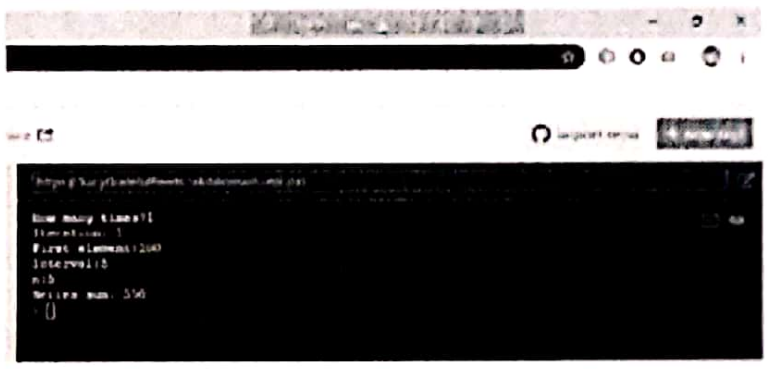


Python Code:

```
def ssum(N,I,F):

  if (N==0):

     return 0
```

```
    elif (N>=1):
        return ssum(N-1,I,F)+F+(N-1)*I

# Main
t=int(input('How many times?'))
for i in range(t):
    print('Iteration:',i+1)
    f=int(input('First element:'))
    d=int(input('Interval:'))
    n=int(input('n:'))
    print('Series sum:', ssum(n,d,f))
```

Sample Input/Output:



QUESTION-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.

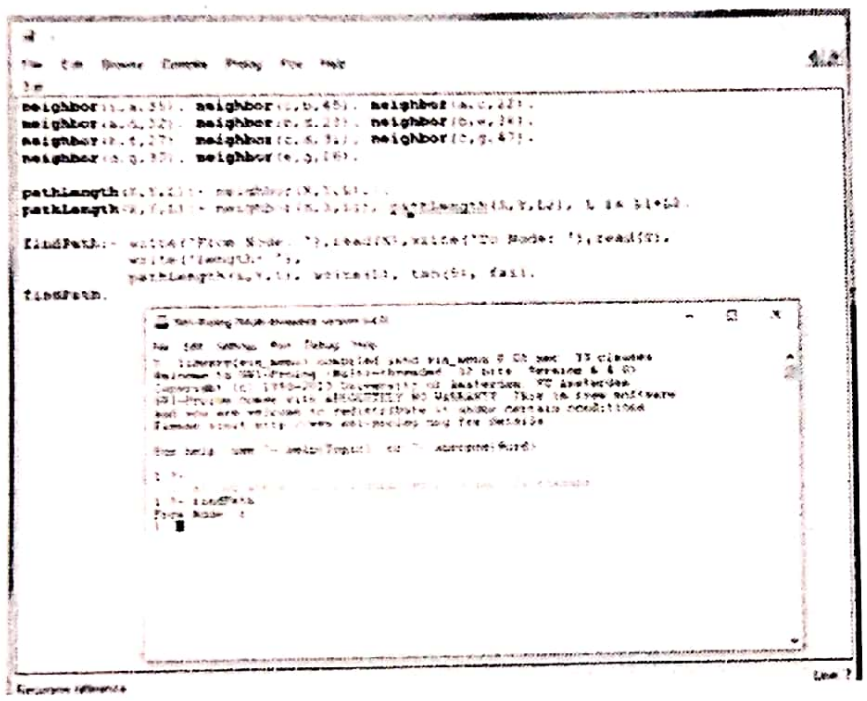
ANSWER:

Prolog Code:

```
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.


findPath:- write('From Node: '),read(X),write('To Node: '),read(Y),

           write('Length: '),

           pathLength(X,Y,L), write(L), tab(5), fail.

findPath.
```

Sample Input/Output:

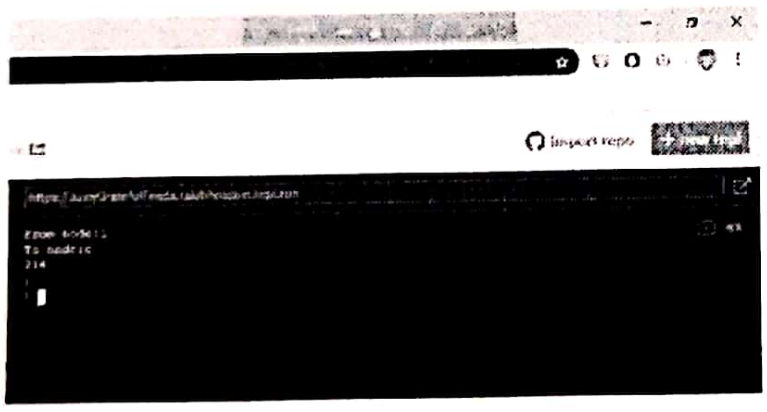


Python Code:

```
neighbour=[('i', 'a', 35),('i', 'b', 45),

   ('a', 'c', 22),('a', 'd', 32),

        ('b', 'd', 28),('b', 'e', 36),('b', 'f', 27),('c', 'd', 31),('c', 'g', 47),

        ('d', 'g', 30),('e', 'g', 26)]

l =0

def pathLength(X,Y):

  i=0

  global l

  while(i <= 10):
```

```
    if(neighbour[i][0] == X):
        l=l+neighbour[i][2]
        if(neighbour[i][1] == Y):
            l = neighbour[i][2]
            break
        pathLength(neighbour[i][1],Y)

    i=i+1
  return l;

# Main
f=str(input('From node:'))
t=str(input('To node:'))
len = pathLength(f,t)
print(len)
```

Sample Input/Output:



QUESTION-3:

Modify the Python and Prolog codes demonstrated above to find h_2

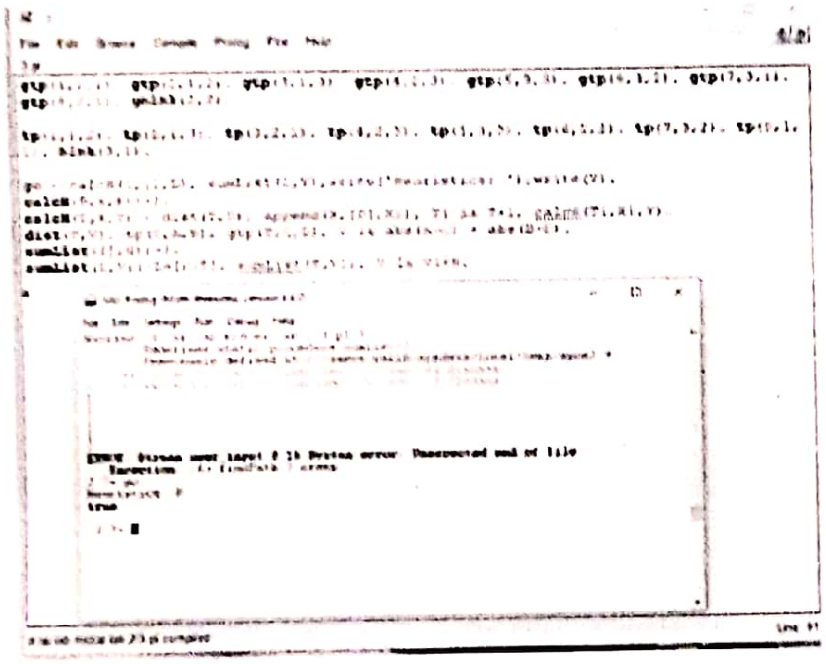
ANSWER:

Prolog Code:

```
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:- 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.
```

Sample Input/Output:



Python Code:

```
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)


# Procedure to find the number of mismatches

i,h=0,0

dis =0
```

```
while(i<=7):
    if ((gtp[i][1] != tp[i][1])|(gtp[i][2] != tp[i][2])):
        dis = abs(tp[i][1]-gtp[i][1]) + abs(tp[i][2]-gtp[i][2])
        h=h+dis


    i=i+1
print('Heuristics 2: ',h)
```

Sample Input/Output: