



Ahsanullah University of Science and Technology (AUST)
Department of Computer Science and Engineering

Course No. : CSE4108

Course Title: Artificial Intelligence Lab

Date of Submission: 30/01/2021

Submitted By:

Lab Group: A1

Section: A

Name: Umme Habiba

ID: 170104004

Submitted To:

Dr. S.M.A Al Mamun

Tonmoy Hossain Dihan

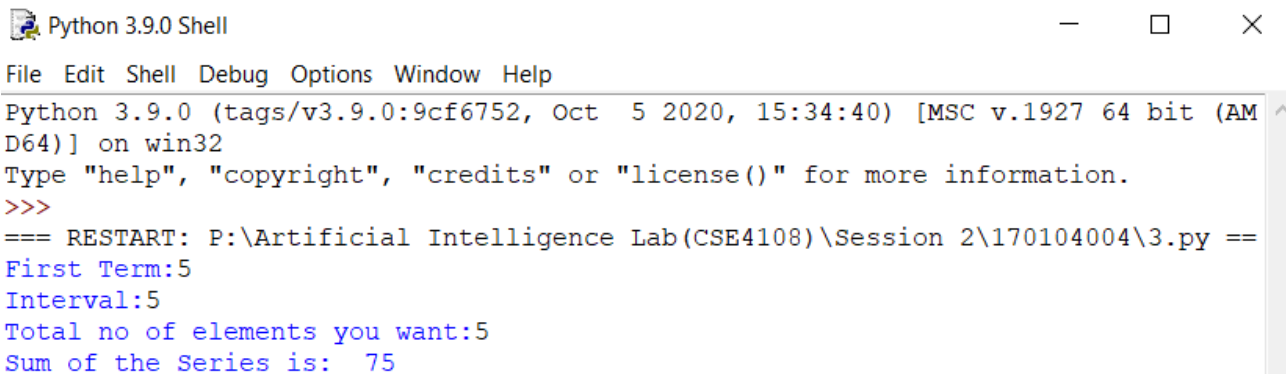
Question-3: Define a recursive procedure in Python and in Prolog to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.

Answer: For Python is given below

Code:

```
firstval=1
i=0
def SumOfSeries(n):
    if (n==0):
        return 0
    x=SumOfSeries(n-1)
    term=firstval+((n-1)*i)
    return x+term
firstval=int(input('First Term:'))
i=int(input('Interval:'))
n=int(input('Total term you want:'))
print('Sum of the Series is: ',SumOfSeries(n))
```

Input/Output:

A screenshot of a Python 3.9.0 Shell window. The window title is "Python 3.9.0 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The status bar at the bottom shows "Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32". The main text area shows the following output:

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: P:\Artificial Intelligence Lab(CSE4108)\Session 2\170104004\3.py ==
First Term:5
Interval:5
Total no of elements you want:5
Sum of the Series is: 75
```

Explanation: Here we have to calculate sum of the series. For that we have taken first term, interval and number of the elements as an input to get the output of the sum of the series.

We know that nth term series is $=(\text{firstval} + ((n-1)*i))$. Here, we have used recursive function .So, when $n=0$ it will return 0 value which is the base case to stop calling the recursion function .To get the total sum , we have used this formula , $\text{total} = \text{SumOfSeries}(n-1) + \text{term}$.

Answer: For Prolog is given below

Code:

```
findSumOfSeries(0,0,_,_):-!.
```

```
findSumOfSeries(N,X,F,I):- N>0 ,N1 is (N-1), findSumOfSeries(N1,Y,F,I),Term  
is F+(N-1)*I,
```

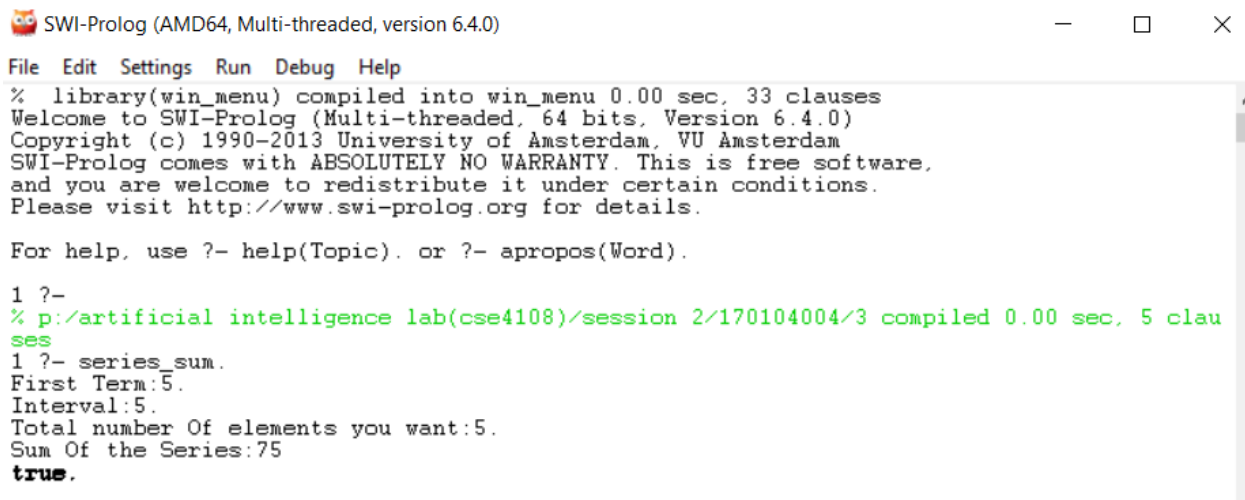
```
X is Y+Term .
```

```
series_sum:-write('First Term:'),read(F),write('Interval:'),read(I),write('Total  
number Of elements you want:'),read(S),write('Sum Of the Series:')
```

```
,findSumOfSeries(S,V,F,I),write(V),tab(5),fail.
```

```
series_sum.
```

Input/Output:



```
SWI-Prolog (AMD64, Multi-threaded, version 6.4.0)
File Edit Settings Run Debug Help
% library(win_menu) compiled into win_menu 0.00 sec, 33 clauses
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.4.0)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
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 ?-
% p:/artificial intelligence lab(cse4108)/session 2/170104004/3 compiled 0.00 sec, 5 clauses
1 ?- series_sum.
First Term:5.
Interval:5.
Total number Of elements you want:5.
Sum Of the Series:75
true.
```

Explanation: In prolog there is no return type , so we have to declare it as a parameter in the function . In the first sentence we have found that, if n is 0 it will return 0 value. In the next term, to get the total sum of the series, we have to count this till $n > 0$ and have to assigned the last term $N-1$ in N . In findSumOfSeries, Y have saved the total sum till $(N-1)$. We have saved the last term in Term ; and X (The total sumOfSeries) will be $Y + \text{Term}$.

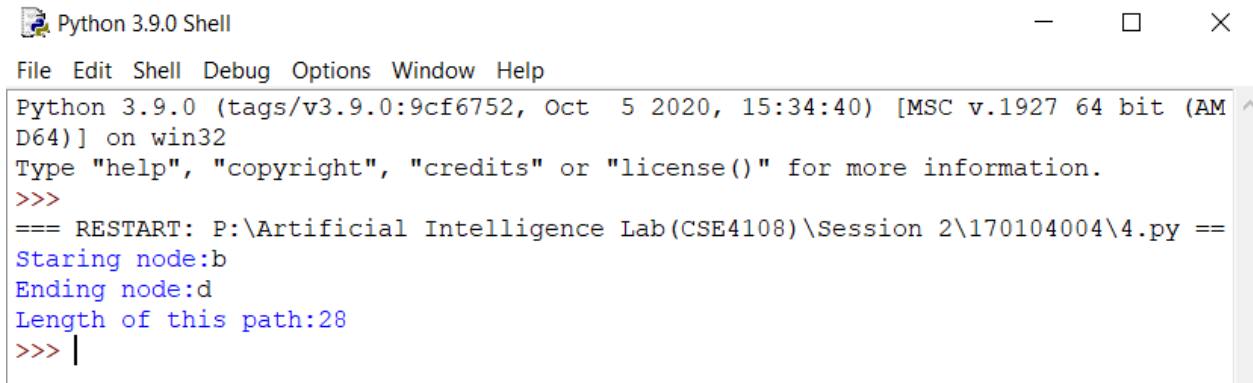
Question-4: Define a recursive procedure in Python and in Prolog to find the length of a path between two vertices of a directed weighted graph.

Answer: For Python is given below

Code:

```
adj=[('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 pathLen(X,Y):
    i=0
    global l
    while(i <= 10):
        if(adj[i][0] == X):
            l=l+adj[i][2]
            if(adj[i][1] == Y):
                l = adj[i][2]
                break
            pathLen(adj[i][1],Y)
        i=i+1
    return l;
s=str(input('Staring node:'))
e=str(input('Ending node:'))
length = pathLen(s,e)
print('Length of this path:',end='')
print(length)
```

Input/Output:



```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: P:\Artificial Intelligence Lab(CSE4108)\Session 2\170104004\4.py ===
Starting node:b
Ending node:d
Length of this path:28
>>> |
```

Explanation: Here we have set the nodes, adjacent nodes and the corresponding weights. Firstly we have defined a recursive function to find our result. We have taken starting and ending nodes as input. Here we use global keyword so that we can modify the variable `l` outside of the scope.

Answer: For Prolog is given below

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('Starting Node: '),read(X),write('Ending Node: '),read(Y),
write('Distance: '),
pathLength(X,Y,L), write(L), tab(5), fail.

findPath.

```

Input/Output:

```

% p:/artificial intelligence lab(cse4108)/session 2/170104004/4 compiled 0.00 sec, 16 cla
uses
[1] 14 ?- findPath.
Starting Node: b.
Ending Node: d.
Length of this path: 28
true.

```

Explanation: Here we have set the nodes, adjacent nodes and the corresponding weights. To find the length, we have used pathLength function . If X is the source and Y is destination, then we find the total length in L. But if X travels to Z which is length L1, then Z travels to Y which is length L2, then the total length will be $L=L1+L2$.

Question-5: Modify the Python and Prolog codes demonstrated above to find h_2 and h_3 discussed above.

Answer: For Python is given below

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 = (2,2)
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)
def H2():
    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=i+1
    print('Heuristics 2: ',h)
position = [[0, 0, 0, 0, 0, 0, 1, 0],
            [0, 0, 0, 1, 0, 0, 0, 0],
            [1, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 0, 0, 0],
            [0, 0, 0, 0, 0, 1, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 0, 0, 1]]
def valid(x, y):
    if (x >= 0 and x < 8 and y >= 0 and y < 8):
        return True
    return False
def H3():
    ans = 0
    for i in range(8):
        for j in range(8):
            if (position[i][j] == 1):
                for k in range(j + 1, 8):
                    if (position[i][k] == 1):
                        ans = ans + 1
                for k in range(i + 1, 8):
                    if (position[k][j] == 1):
                        ans = ans + 1
            for d in range(1, 8):
                x = i + d
                y = j + d
                if (valid(x, y) and position[x][y] == 1):
                    ans = ans + 1
                x = i - d
                y = j + d
                if (valid(x, y) and position[x][y] == 1):

```

```

        ans = ans + 1
    print('Heuristics 3 :',ans)
H2()
H3()

```

Input/Output:

```

=== RESTART: P:\Artificial Intelligence Lab(CSE4108)\Session 2\170104004\5.py ==
Heuristics 2:  8
Heuristics 3 : 5
>>> |

```

Explanation: To find the h2, first we have checked if the current gtp position matches with the selected gtp .If not then we have find out the distance $\text{abs}(x1-x2)+\text{abs}(y1-y2)$. Through this we have find out, h2 values. To find out h3, first we have set position for queen. If there is a queen we have set it in 1 , and the others values are 0 .Then we have fix the row and check the columns if we find any queen and then we have fix the column and check the rows . After that we have checked it through diagonally if there is any queen left or not. If we find any queen, we have counted the total numbers of queen. Following this, we have found out the h3 value.

Answer: For Prolog is given below

Code:

%pos(i, x, y) is position of queen i is x, y

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

% check(x, y, v) set v is 1 if there is a queen in position x, y


```

% otherwise 0

check(X, Y, V) :- pos(_, X, Y), V is 1, !.

check(_, _, 0) :-!.

% right(X, Y, V) set V = how many queen are there in its right side

right(_, 9, 0):-!.

right(X, Y, V) :- Y1 is Y + 1, right(X, Y1, V1), check(X, Y, V2), V is V1 + V2, !.

% countR(N, A) set A = how many other queen is at queen N's right

countR(N, A) :- pos(N, X, Y), Y1 is Y + 1, right(X, Y1, Ans), A is Ans, !.

up(9, _, 0):-!.

up(X, Y, V):- X1 is X + 1, up(X1, Y, V1), check(X, Y, V2), V is V1 + V2, !.

% similar in upword

countU(N, A) :- pos(N, X, Y), X1 is X + 1, up(X1, Y, Ans), A is Ans, !.

% diagonal up

countDU(N, A) :- pos(N, X, Y), X1 is X + 1, Y1 is Y + 1, digup(X1, Y1, Ans), A is
Ans, !.

%diagonal down

countDD(N, A):- pos(N, X, Y), X1 is X - 1, Y1 is Y + 1, digdown(X1, Y1, Ans), A is
Ans, !.

digup(_, 9, 0):-!.

digup(9, _, 0):-!.

digup(X, Y, V) :- X1 is X + 1, Y1 is Y + 1, digup(X1, Y1, V1), check(X, Y, V2), V is

```

V1 + V2, !.

digdown(_, 9, 0):-!.

digdown(9, _, 0):-!.

digdown(X, Y, V):- X1 is X - 1, Y1 is Y + 1, digdown(X1, Y1, V1), check(X, Y, V2), V is V1 + V2, !.

queen(9, 0):-!.

queen(N, V):- N1 is N + 1, queen(N1, V1), countR(N, A1), countU(N, A2),
countDU(N, A3), countDD(N, A4), V is V1 + A1 + A2 + A3 + A4, !.

h3:- queen(1, V), write('Heuristic 3 = '), write(V), !.

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('Heuristics2: '),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.

Input/Output:

```
% p:/artificial intelligence lab(cse4108)/session 2/170104004/5 compiled 0.00 sec, 51 cla
uses
10 ?- h2.
Heuristic 2 = 8
true.

11 ?- h3.
Heuristic 3 = 5
true.
```

Explanation: In h2, we have checked if it returns 0, that means the value is in the position. But if v is not in it's position then we will return 1 value .In calH we have counted the total value .