**Name: Mohammad Zain Abbas**

**Reg #: 6865**

**DE-36 (CE), Syndicate: A**

**LAB 3 JOURNEL**

**Equipment Used:** Notebook Computer, Python IDLE 3.6

**Lab Tasks:**

1. Implement the undirected Graph 1 and 2 in Python. Show the connectivity as well as the degree of each node within these graphs.

**SOLUTION CODE:**

def get\_Connectivity(Graph):

for i in Graph:

print(i,Graph[i]);

def get\_Degree(Graph):

for i in Graph:

print(i,'Degree = ',len(Graph[i]));

def main():

Graph1={'6':['4'],

'4':['5','3','6'],

'5':['4','2','1'],

'3':['4','2'],

'2':['3','5','1'],

'1':['5','2']}

Graph2={'E':['A','B'],

'A':['E','B','D'],

'B':['A','E','D'],

'D':['A','B','C'],

'C':['D']

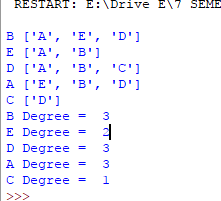
}

get\_Connectivity(Graph2);

get\_Degree(Graph2);

main();

**OUTPUT:**



2. Implement the directed Graph 3 and 4 in Python. Show the connectivity, indegree and outdegree of each node within these graphs.

**SOLUTION CODE:**

def get\_Connectivity(Graph):

for i in Graph:

print(i,Graph[i]);

def get\_outDegree(Graph):

for i in Graph:

print(i,'Degree = ',len(Graph[i]));

def get\_InDegree(Graph):

for i in Graph:

count=0;

for j in Graph:

if(i!=j):

if(i in Graph[j]):

count=count+1;

print(i,"InDegree = ",count);

def main():

Graph3={'7':['11','8'],

'5':['11'],

'3':['8','10'],

'11':['2','9','10'],

'8':['9'],

'2':[],

'9':[],

'10':[],

}

Graph4={'A':['B'],

'B':['C','D','E'],

'C':['E'],

'E':['F'],

'G':['D'],

'D':['E'],

'F':[]

}

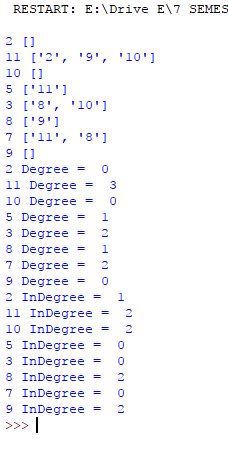
get\_Connectivity(Graph3);

get\_outDegree(Graph3);

get\_InDegree(Graph3);

main();

**OUTPUT:**



**3.** Write a method to find any path between node 6 to node 1 in Graph 1.

**4.** Write a method to find any path between node E to node C in Graph 2.

**5.** Write a method to find any path between node 7 to node 9 in Graph 3.

**6.** Write a method to find any path between node A to node F in Graph 4.

**SOLUTION CODE:**

def path\_Finder(Graph,Snode,Enode,path=[]):

path=path+[Snode];

if (Snode==Enode):

return path;

if (Snode not in Graph):

return None;

for node in Graph[Snode]:

if node not in path:

path=path\_Finder(Graph,node,Enode,path);

if path:

return path;

return None;

def main():

Graph1={'6':['4'],

'4':['3','5','6'],

'5':['1','2','4'],

'3':['2','4'],

'2':['1','3','5'],

'1':['2','5']

}

Graph2={

'A':['B','D','E'],

'B':['A','D','E'],

'C':['D'],

'D':['A','B','C'],

'E':['A','B']

}

Graph3={'7':['8','11'],

'5':['11'],

'3':['8','10'],

'11':['2','9','10'],

'8':['9'],

'2':[],

'9':[],

'10':[],

}

Graph4={'A':['B'],

'B':['C','D','E'],

'C':['E'],

'E':['F'],

'G':['D'],

'D':['E'],

'F':[]

}

print(path\_Finder(Graph1,'6','1'));

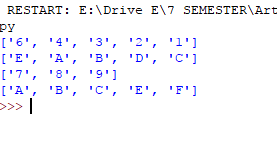
print(path\_Finder(Graph2,'E','C'));

print(path\_Finder(Graph3,'7','9'));

print(path\_Finder(Graph4,'A','F'));

main();

**OUTPUT:**



**7.** Modify Question # 3 to show all possible paths between node 6 to node 1 in Graph 1.

**8.** Modify Question # 4 to show all possible paths between node E to node C in Graph 2.

**9.** Modify Question # 5 to show all possible paths between node 7 to node 9 in Graph 3.

**10.**Modify Question # 6 to show all possible paths between node A to node F in Graph 1.

**SOLUTION CODE:**

def All\_path\_Finder(Graph,Snode,Enode,path=[]):

path=path+[Snode];

if (Snode==Enode):

return [path];

if (Snode not in Graph):

return None;

paths=[];

for node in Graph[Snode]:

if node not in path:

new\_path=All\_path\_Finder(Graph,node,Enode,path);

for p in new\_path:

paths.append(p);

return paths;

def main():

Graph1={'6':['4'],

'4':['3','5','6'],

'5':['1','2','4'],

'3':['2','4'],

'2':['1','3','5'],

'1':['2','5']

}

Graph2={

'A':['B','D','E'],

'B':['A','D','E'],

'C':['D'],

'D':['A','B','C'],

'E':['A','B']

}

Graph3={'7':['8','11'],

'5':['11'],

'3':['8','10'],

'11':['2','9','10'],

'8':['9'],

'2':[],

'9':[],

'10':[],

}

Graph4={'A':['B'],

'B':['C','D','E'],

'C':['E'],

'E':['F'],

'G':['D'],

'D':['E'],

'F':[]

}

print("ALL PATHS from 6 - 1=",All\_path\_Finder(Graph1,'6','1'));

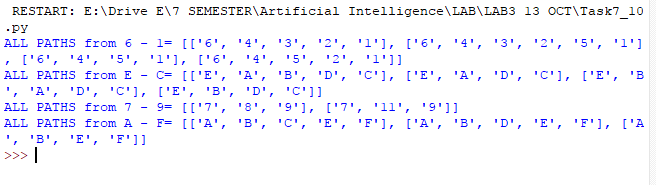
print("ALL PATHS from E - C=",All\_path\_Finder(Graph2,'E','C'));

print("ALL PATHS from 7 - 9=",All\_path\_Finder(Graph3,'7','9'));

print("ALL PATHS from A - F=",All\_path\_Finder(Graph4,'A','F'));

main();

**OUTPUT:**



Q: Suppose you have been given with a following 3x3 3-bit grayscale image. You job is to decompose it into an undirected graph where each pixel within an image represent a nod and adjacent nodes are connected to each other via 4-connectivity pattern. Show all possible

paths between pixel 150 and pixel 165.

|  |  |  |
| --- | --- | --- |
| 150 | 2 | 5 |
| 80 | 145 | 45 |
| 74 | 102 | 165 |

**SOLUTION CODE:**

def All\_path\_Finder(Graph,Snode,Enode,path=[]):

path=path+[Snode];

if (Snode==Enode):

return [path];

if (Snode not in Graph):

return None;

paths=[];

for node in Graph[Snode]:

if node not in path:

new\_path=All\_path\_Finder(Graph,node,Enode,path);

for p in new\_path:

paths.append(p);

return paths;

def main():

Graph\_Image={'2':['5','145','150'],

'5':['2','45'],

'45':['5','145','165'],

'74':['80','102'],

'80':['74','145','150'],

'102':['74','145','165'],

'145':['2','45','80','102'],

'150':['2','80'],

'165':['45','102']

}

print("ALL PATHS from 150 - 165=",All\_path\_Finder(Graph\_Image,'150','165'));

main();

**OUTPUT:**

