**Assignment # 1**

**NAME: Saad Ahmed Khan**

**COURSE:Discrete Mathematics**

**STUDENT ID: 64050**

**COURSE ID: 103348**

**Code:**

from collections importdefaultdict

class Graph:

def \_\_init\_\_(self,vertices

self.V= vertices

self.graph= defaultdict(list)

# To store transitive closure

self.tc = [[0for j inrange(self.V)] foriinrange(self.V)]

defaddEdge(self,u,v):

self.graph[u].append(v)

defDFSUtil(self,s,visited, matrix):

visited[s] = True

for c inself.graph[s]:

matrix[s][c] = 1

if visited[c] == False:

self.DFSUtil(c, visited, matrix)

elif visited[c] == True:

foriinrange(self.V):

if matrix[s][i] == 1:

matrix[c][i] = 1

foriinrange(self.V):

if matrix[c][i] == 1:

matrix[s][i] = 1

return

deftransitiveClosure(self):

matrix = [[0for j inrange(self.V)] foriinrange(self.V)]

visited = [False]\*self.V

for v inself.graph.keys():

if visited[v] == False:

self.DFSUtil(v, visited, matrix)

return matrix

# Create a graph given in the above diagram

g = Graph(4)

g.addEdge(2, 0)

g.addEdge(0, 2)

g.addEdge(3, 1)

g.addEdge(1, 3)

g.addEdge(2, 3)

g.addEdge(3, 2)

print"Transitive closure matrix is"

printg.transitiveClosure();

**Output:**

