tree = {

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

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

'C':['F','G'],

'D':[],

'E':[],

'F':[],

'G':[]

}

start = input("Enter the start node : ")

def dfs\_goalsearch(tree):

goal = input("Enter Goal Node : ")

visited=[]

stack = [[start]] #list of list for best possible path

if start==goal:

print("Start is the Goal node")

return start

while stack:

path = stack.pop() #path is the variable that is taking a path from list of list

node = path[-1] #poping last element in stack

if node not in visited:

visited.append(node)

neigbour=tree[node]

for i in neigbour:

new\_path=list(path) #creating a list using list() constructor

new\_path.append(i)

stack.append(new\_path)

if i==goal:

return new\_path

def dfs\_traversal(tree):

visited = []

stack = [start] # stack has [A]

while stack : # while stack is nnot empty

node=stack.pop() #poping the first element in stack

if node not in visited :

visited.append(node) #adding the node which has been visited

neighbour=tree[node] #taking the childern/neigbours of node variable and putting in variable neigbour

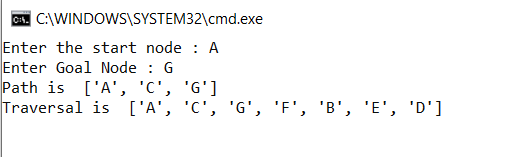
for i in neighbour :

stack.append(i) #putting the neigbours in the stack

return visited

print("Path is ",dfs\_goalsearch(tree))

print("Traversal is ",dfs\_traversal(tree))



tree ={ 'A':['B','C'],

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

'C':['F','G'],

'D':[], 'E':[],'F':[],'G':[]

}

start=input("enter start node:")

goal=input("enter goal node:")

max\_depth=int(input("enter max depth:"))

path=[]

level=0;

def depth\_limited\_search(start,goal,path,level,max\_depth):

print("Current level is ",level)

path.append(start)

if start==goal:

print("Goal node is found")

return path

if level==max\_depth:

return False

print("Expanding current node :",start)

neighbour=tree[start]

for i in neighbour :

if depth\_limited\_search(i, goal, path, level+1, max\_depth):

return path

print(path.pop())

return False

result=depth\_limited\_search(start, goal, path, level, max\_depth)

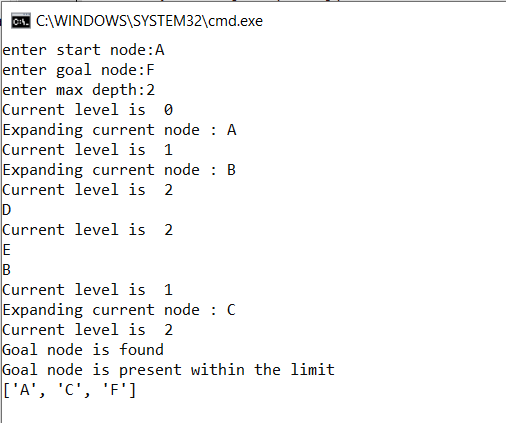
if result:

print("Goal node is present within the limit ");

print(path)

else :

print("Goal node is not present within the max limit ")



tree ={ 'A':['B','C'],

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

'C':['F','G'],

'D':[], 'E':[],'F':[],'G':[]

}

start=input("enter start node:")

goal=input("enter goal node:")

max\_depth=int(input("enter max depth:"))

path=[]

level=0;

def iterative\_dfs(start,goal,path,level,max\_depth):

print("Current level is ",level)

path.append(start)

if start==goal:

print("Goal node is found")

return path

if level==max\_depth:

return False

print("Expanding current node :",start)

neighbour=tree[start]

for i in neighbour :

if iterative\_dfs(i, goal, path, level+1, max\_depth):

return path

print(path.pop())

return False

for i in range(max\_depth+1):

path=[]

result=iterative\_dfs(start, goal, path, level, i)

if result:

print("Goal node is present within the limit ");

print(path)

break

else :

print("Goal node is not present within the max limit ")

