tree = {

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

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

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

'D':[],

'E':[],

'F':[],

'G':[]

}

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

def bfs\_goalsearch(tree):

goal = input("Enter Goal Node : ")

visited = []

queue = [start]

if start == goal:

print("Start itself is the goal node")

return start

visited.append(start)

while queue :

node=queue.pop(0)

neighbour=tree[node]

for i in neighbour:

queue.append(i)

visited.append(i)

if i==goal:

return visited

def bfs\_traversal(tree):

visited = []

queue = [start] # queue has [A]

while queue : # while queue is nnot empty

node=queue.pop(0) #poping the first element in queue

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 :

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

return visited

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

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

