**Lab 3: To study and implement Graph search algorithms in Python**

1. Provide the implementation of DFS and BFS algorithms in Python.

**BFS**

g={}  
g[1]=[3,4]  
g[2]=[1,4]  
g[3]=[]  
g[4]=[3]  
print(g)  
n=int(input(**'enter the node'**))  
ind=0  
**for** k **in** g.keys():  
 **if** n **in** g[k]:  
 ind+=1  
print(ind)  
print(len(g[n]))  
stack=[]  
**def** push(s,e):  
 **if** e **in** s:  
 **return** s.insert(0,e)  
 **return** s  
**def** pop(s):  
 **if** (len(s)==0):  
 **return  
 else**:  
 **return** s.pop(0)  
**def** empty(s):  
 **return** len(s)==0  
push(stack,1)  
**while** empty(stack)== **False**:  
 e=pop(stack)  
 print(e)  
 **for** k **in** g[e]:  
 push(stack,k)

**DFS:**

g={}  
g[1]=[3,4]  
g[2]=[1,4]  
g[3]=[]  
g[4]=[3]  
print(g)  
n=int(input(**'enter the node'**))  
ind=0  
**for** k **in** g.keys():  
 **if** n **in** g[k]:  
 ind+=1  
print(ind)  
print(len(g[n]))  
queue=[]  
**def** enqueue(q,e):  
 **if** e **in** q:  
 **return** q.append(e)  
  
**def** dequeue(q):  
 **if** (len(q)==0):  
 **return  
 else**:  
 **return** q.pop(0)  
**def** empty(q):  
 **return** len(q)==0  
enqueue(queue,1)  
output=[]  
**while** empty(queue)== **False**:  
 e=dequeue(queue)  
 **if** e **in** output: **continue** output.append(e)  
 **for** k **in** g[e]:  
 enqueue(queue,e)  
print(output)