ASS 1

```
# Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected
# graph and develop a recursive algorithm for searching all the vertices of a graph or tree data
# structure.

import sys

visited = []
queue = []

def bfs(visited, graph, node,searchNodee):
  print("BFS: ",end='')
  visited.append(node)
  queue.append(node)

  while queue:
    m = queue.pop(0)
    print (m, end = " ")
    #v
    if(m == searchNodee):
       break



    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

dfsVisited = set()
```

```
def dfs(dfsVisited, graph, node, searchNodee):
    if node not in dfsVisited:
        #1
        print (node, end=' ')
        if(node == searchNodee):
          #a
          sys.exit()
        dfsVisited.add(node)
        for neighbour in graph[node]:
            dfs(dfsVisited, graph, neighbour, searchNodee) #c


graph = {}


while True:
    root = input("Enter the root node: [input/done]  ")


    if(root=="done"):
        break


    if root not in graph:
        graph[root] = []


        while True:
            #k
            child = input("Please enter the child nodes of "+root+": [Enter input/done]  ")


            if child == "done":
                print()
                break


            if child not in graph[root]:
```

```
        graph[root].append(child)


print("\nThe graph is:\n")
print(graph, end='\n\n')


searchNode = input("Enter the node you want to search: ",)
print()
first_key = next(iter(graph))
bfs(visited, graph, first_key, searchNode)
print()
print("DFS: ",end='')
dfs(dfsVisited, graph, first_key, searchNode)


# enter the input in following way:
# a
# b
# c
# done
# b
# d
# e
# done
# c
# f
# g
# done
# d
# done
# e
# done
# f
```

done

g

done

done

c

ASS 1 done