**DFS**

t = {}

def addEdge(u, v):

if u not in t:

t[u] = []

t[u].append(v)

def dfs(start, search):

stack = [start]

path = []

found = False

while stack:

c = stack.pop()

path.append(c)

if c == search:

found = True

break

if c in t:

for i in reversed(t[c]):

stack.append(i)

if found:

print("Path as follows:")

print(" -> ".join(path))

print("Node found")

else:

print("Node not found")

def tree(root):

levels = {}

queue = [(root, 0)]

while queue:

node, level = queue.pop(0)

if level not in levels:

levels[level] = []

levels[level].append(node)

if node in t:

for child in t[node]:

queue.append((child, level + 1))

max\_level = max(levels.keys())

width = 2 \*\* (max\_level + 1)

for level in range(max\_level + 1):

nodes = levels.get(level, [])

line = ' '.join(node.center(width // (2 \*\* level)) for node in nodes)

print(line)

while True:

try:

n = int(input("Enter no.of nodes:"))

if n < 0:

print("Invalid input. Please enter a positive number.")

elif n == 0:

print("Invalid input. Please enter a number greater than zero.")

else:

break

except ValueError:

print("Invalid input. Please enter an integer.")

print("Enter nodes along with their connections (in parent-child format):")

start = None

for x in range(n - 1):

while True:

try:

parent, child = input().split()

if start is None:

start = parent

addEdge(parent, child)

break

except ValueError:

print("Invalid format! Please enter parent-child pair in the same line.")

print("Tree is:")

tree(start)

search = input("Enter the element to be searched: ")

dfs(start, search)