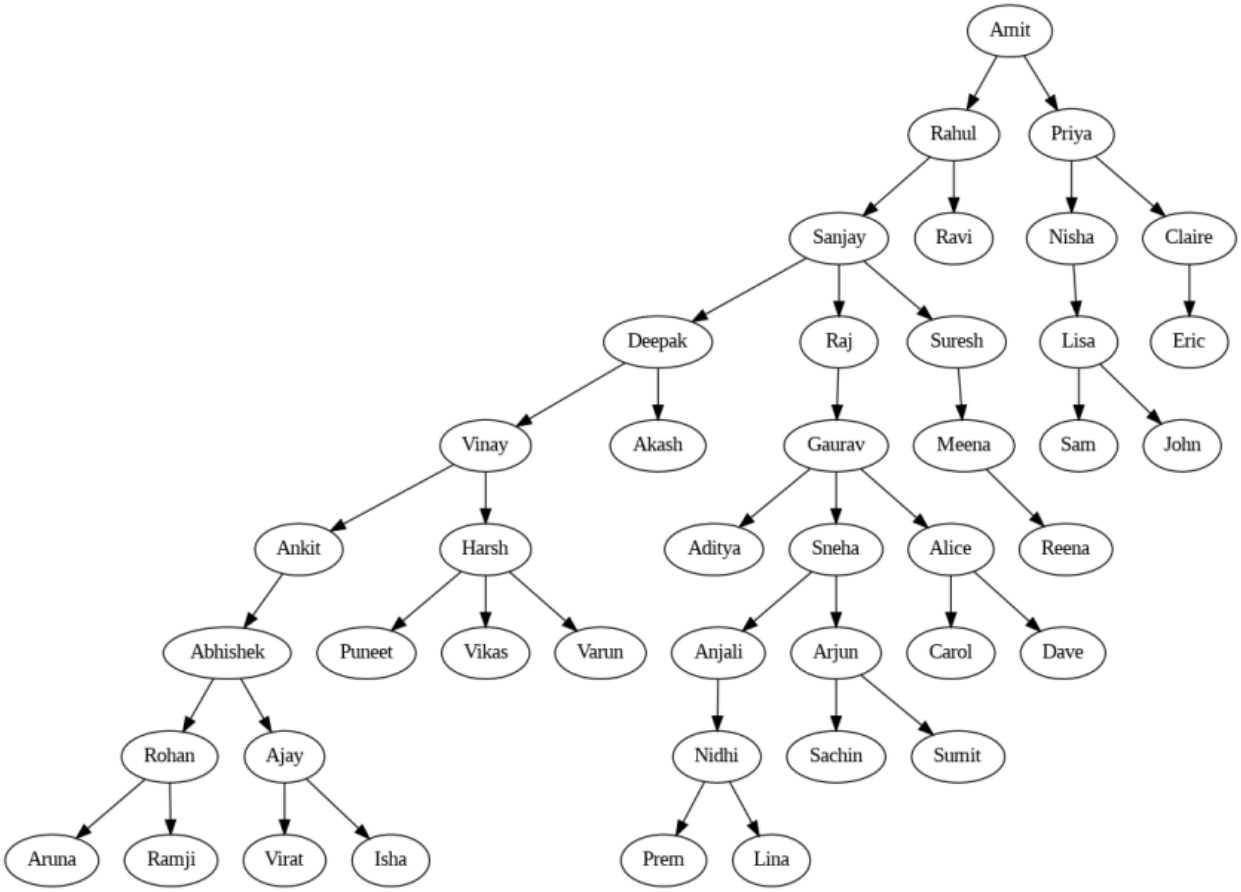
PRACTICAL - 1

AIM: Write a python program to implement Breadth first search (BFS) Traversal on input tree.

Search String = Prem
Path: Amit->Rahul->Sanjay->Raj->Gaurav->Sneha->Anjali->Nidhi->Prem
Path Cost=8
Input Tree:



Code:

```
class Node:
  def __init__(self, name,Parent, Child= None):
    self.name=name
    self.parent=Parent
    self.child =[]
    self.level=0
    if(self.parent != None):
      self.level = self.parent.level+1
```

```
  def insertChild(self, Node):
    self.child.append(Node)

class Tree:
  def __init__(self, root):
    self.root=root

  def insertNode (self, name,Parent):
    node=Node(name,Parent)
    Parent.insertChild(node)
    return node


root= Node("Amit",None)
tree=Tree(root)

Rahul = tree.insertNode("Rahul",root)
Priya = tree.insertNode("Priya",root)
Sanjay = tree.insertNode("Sanjay",Rahul)
Ravi = tree.insertNode("Ravi",Rahul)
Nisha = tree.insertNode("Nisha",Priya)
Claire = tree.insertNode("Claire",Priya)
Deepak = tree.insertNode("Deepak",Sanjay)
Raj = tree.insertNode("Raj",Sanjay)
Suresh = tree.insertNode("Suresh",Sanjay)
Lisa = tree.insertNode("Lisa",Nisha)
Eric = tree.insertNode("Eric",Claire)
Vinay = tree.insertNode("Vinay",Deepak)
Akash = tree.insertNode("Akash",Deepak)
Sam = tree.insertNode("Sam",Lisa)
Jhon = tree.insertNode("Jone",Lisa)
Gaurav = tree.insertNode("Gaurav",Raj)
Meena = tree.insertNode("Meena",Suresh)
Ankit = tree.insertNode("Ankit",Vinay)
Aditya = tree.insertNode("Aditya",Gaurav)
Sneha = tree.insertNode("Sneha",Gaurav)
Alice = tree.insertNode("Alice",Gaurav)
Reena = tree.insertNode("Reena",Meena)
Abhishek = tree.insertNode("Abhishek",Ankit)
Harsh = tree.insertNode("Harsh",Vinay)
Puneet = tree.insertNode("Puneet",Harsh)
Vikas = tree.insertNode("Vikas",Harsh)
```

```
Varun = tree.insertNode("Varun",Harsh)
Anjali = tree.insertNode("Anjali",Sneha)
Arjun = tree.insertNode("Arjun",Sneha)
carol = tree.insertNode("carol",Alice)
Dave = tree.insertNode("Dave",Alice)
Rohan = tree.insertNode("Rohan",Abhishek)
Ajay = tree.insertNode("Ajay",Abhishek)
Aruna = tree.insertNode("Aruna",Rohan)
Ramji = tree.insertNode("Ramji",Rohan)
Virat = tree.insertNode("Virat",Ajay)
Isha = tree.insertNode("Isha",Ajay)
Nidhi = tree.insertNode("Nidhi",Anjali)
Prem = tree.insertNode("Prem",Nidhi)
Lina = tree.insertNode("Lina",Nidhi)
Sachin = tree.insertNode("Sachin",Arjun)
Sumit = tree.insertNode("Sumit",Arjun)


def bfs_search(searchValue, fullTree):
    queue = []
    queue.append(fullTree.root)
    solutionNode = None

    while queue:
        tempNode = queue.pop(0)

        if tempNode.name == searchValue:
            solutionNode = tempNode
            break

        queue.extend(tempNode.child)

    return solutionNode

def drawPath(solNode):
    list1 = []
    tempNode = solNode

    while tempNode is not None:
        list1.append(tempNode.name)
        tempNode = tempNode.parent

    list1.reverse()
```

```
    print("Path:")
    print("->".join(list1))
    print("Path Cost =", len(list1) - 1)
target = "Prem"  # The target node
solutionNode = bfs_search(target, tree)

if solutionNode:
    drawPath(solutionNode)
else:
    print("Node not found in the tree")
```

Output:

```
Path:
Amit->Rahul->Sanjay->Raj->Gaurav->Sneha->Anjali->Nidhi->Prem
Path Cost = 8
```