TREES AND GRAPHS

TREES

```
class node:
    def __init__(self,data):
        self.left=None
        self.data=data
        self.right=None
def inorder(root):
    if root:
        inorder(root.left)
        print(root.data,end=" ")
        inorder(root.right)
def preorder(root):
    if root:
        print(root.data,end=" ")
        preorder(root.left)
        preorder(root.right)
def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(root.data,end=" ")
r=node(1)
r.left=node(2)
r.right=node(3)
r.left.left=node(4)
r.right.right=node(5)
inorder(r)
print( )
preorder(r)
```

```
print( )
postorder(r)
```

GRAPHS

```
class Graph:
    def __init__(self):
        self.matrix=[[0]*5 for i in range(5)]
        print(self.matrix)
    def addvertex(self,a,b):
        self.matrix[a][b]=1
    def print(self):
        for i in self.matrix:
            print(i)
g=Graph()
g.addvertex(1,2)
g.addvertex(4,2)
g.addvertex(1,4)
g.addvertex(2,3)
g.addvertex(4,3)
g.print()
```

BFS

```
class Graph:
    def __init__(self):
        self.matrix=[[0]*5 for i in range(5)]
        print(self.matrix)
    def addvertex(self,a,b):
        if a not in self.matrix:
            self.matrix[a]=[b]
        else:
            self.matrix[a].append(b)
    def print(self):
```

```
        for i in self.matrix:
            print(i)
    def bfs(self,data):
        visited=[]
        queue=[data]
        while queue:
            vertex=queue.pop(0)
            print(vertex)
            if vertex in self.matrix:
                for i in self.matrix[vertex]:
                    if i not in visited:
                        visited.append(i)
                        queue.append(i)
g=Graph()
g.addvertex(1,2)
g.addvertex(4,2)
g.addvertex(1,4)
g.addvertex(2,3)
g.addvertex(4,3)
g.bfs(1)
g.print()
```

DFS

```
class Graph:
    def __init__(self):
        self.matrix=[[0]*5 for i in range(5)]
        print(self.matrix)
    def addvertex(self,a,b):
        if a not in self.matrix:
            self.matrix[a]=[b]
        else:
            self.matrix[a].append(b)
    def print(self):
```

```
        for i in self.matrix:
            print(i)
    def dfs(self,data):
        visited=[]
        queue=[data]
        while queue:
            vertex=queue.pop()
            print(vertex)
            if vertex in self.matrix:
                for i in self.matrix[vertex]:
                    if i not in visited:
                        visited.append(i)
                        queue.append(i)
g=Graph()
g.addvertex(1,2)
g.addvertex(4,2)
g.addvertex(1,4)
g.addvertex(2,3)
g.addvertex(4,3)
g.dfs(1)
g.print()
```