Making binary tree manually

In [1]:

```
def first_tree():
    root=Node(10,None,None,None)
    n1=Node(20,None,None,None)
    n2=Node(30,None,None,None)
    n3=Node(40,None,None,None)
    n4=Node(50,None,None,None)
    n5=Node(70,None,None,None)
    n6=Node(80,None,None,None)
    n7=Node(90,None,None,None)
    n8=Node(100,None,None,None)
    root.left=n1
    root.right=n2

    n1.left=n3
    n3.parent=n1
    n1.right=n4
    n4.parent=n1
    n1.parent=root

    n2.left=n5
    n5.parent=n2
    n2.right=n6
    n6.parent=n2
    n2.parent=root

    n6.left=n7
    n7.parent=n6
    n6.right=n8
    n8.parent=n6
    return root
def second_tree():
    root=Node(100,None,None,None)
    n1=Node(200,None,None,None)
    n2=Node(300,None,None,None)
    n3=Node(400,None,None,None)
    n4=Node(500,None,None,None)
    n5=Node(700,None,None,None)
    n6=Node(800,None,None,None)
    n7=Node(900,None,None,None)
    n8=Node(1000,None,None,None)
    root.left=n1
    root.right=n2

    n1.left=n3
    n3.parent=n1
    n1.right=n4
    n4.parent=n1
    n1.parent=root

    n2.left=n5
    n5.parent=n2
    n2.right=n6
    n6.parent=n2
    n2.parent=root

    n6.left=n7
```

```
    n7.parent=n6
    n6.right=n8
    n8.parent=n6
    return root
```

In [39]:

```
#obj binary tree
#         10
#       /   \
#     20    30
#    / \   /  \
#   40 50 70  80
#            /  \
#           90  100

#obj2 binary tree
#         100
#        /   \
#     200    300
#    /  \    /  \
#   400 500 700  800
#            /   \
#           900  1000
```

In [2]:

```
class Node:
    def __init__(self,element, left , parent, right):
        self.element=element
        self.left=left
        self.parent=parent
        self.right=right
class binary_tree:
    def __init__(self,root):
        self.root=root
    def height(self,root):
        if root== None:
            return -1
        else:
            a,b=(self.height (root.left), self.height(root.right))
            if a>b:
                return 1+ a
            else:
                return 1+b
    def level(self,root):
        if root.parent== None:
            return 1
        else:
            return 1+self.level(root.parent)
    def preorder(self,root):
        if root !=None:
            print(root.element,end=" ")
            self.preorder(root.left)
            self.preorder(root.right)
        else:
            return
    def inorder(self,root):
        if root !=None:
            self.inorder(root.left)
            print(root.element,end=" ")
            self.inorder(root.right)
        else:
            return
    def postorder(self,root):
        if root !=None:
            self.postorder(root.left)
            self.postorder(root.right)
            print(root.element,end=" ")
        else:
            return
    def same_or_not(self,root1,root2):
        if root1 !=None:
            if root1.element==root2.element:
                self.same_or_not(root1.left,root2.left)
                self.same_or_not(root1.right,root2.right)

            else:
                return "\nNot Same"

        return "\nSame"
    def copy_bt(self,root):
        obj4=binary_tree(root)
```

```
        return obj4
root=first_tree()
root2=second_tree()
root3=first_tree()
obj=binary_tree(root)
obj2=binary_tree(root2)
obj3=binary_tree(root3)
print("Height :",obj.height(obj.root))
print("Level  :",obj.level(obj.root.left.right))
print("Level  :",obj.level(obj.root.right.right.right))
print("Pre-Order")
obj.preorder(obj.root)
print("\nIN-Order")
obj.inorder(obj.root)
print("\npost-Order")
obj.postorder(obj.root)
print(obj.same_or_not(obj.root,obj2.root))
print(obj.same_or_not(obj.root,obj3.root))
obj4=obj.copy_bt(obj.root)
print("Copy of Binary Tree:")
obj4.preorder(obj4.root)
print("\nMain Binary Tree")
obj.preorder(obj.root)
```

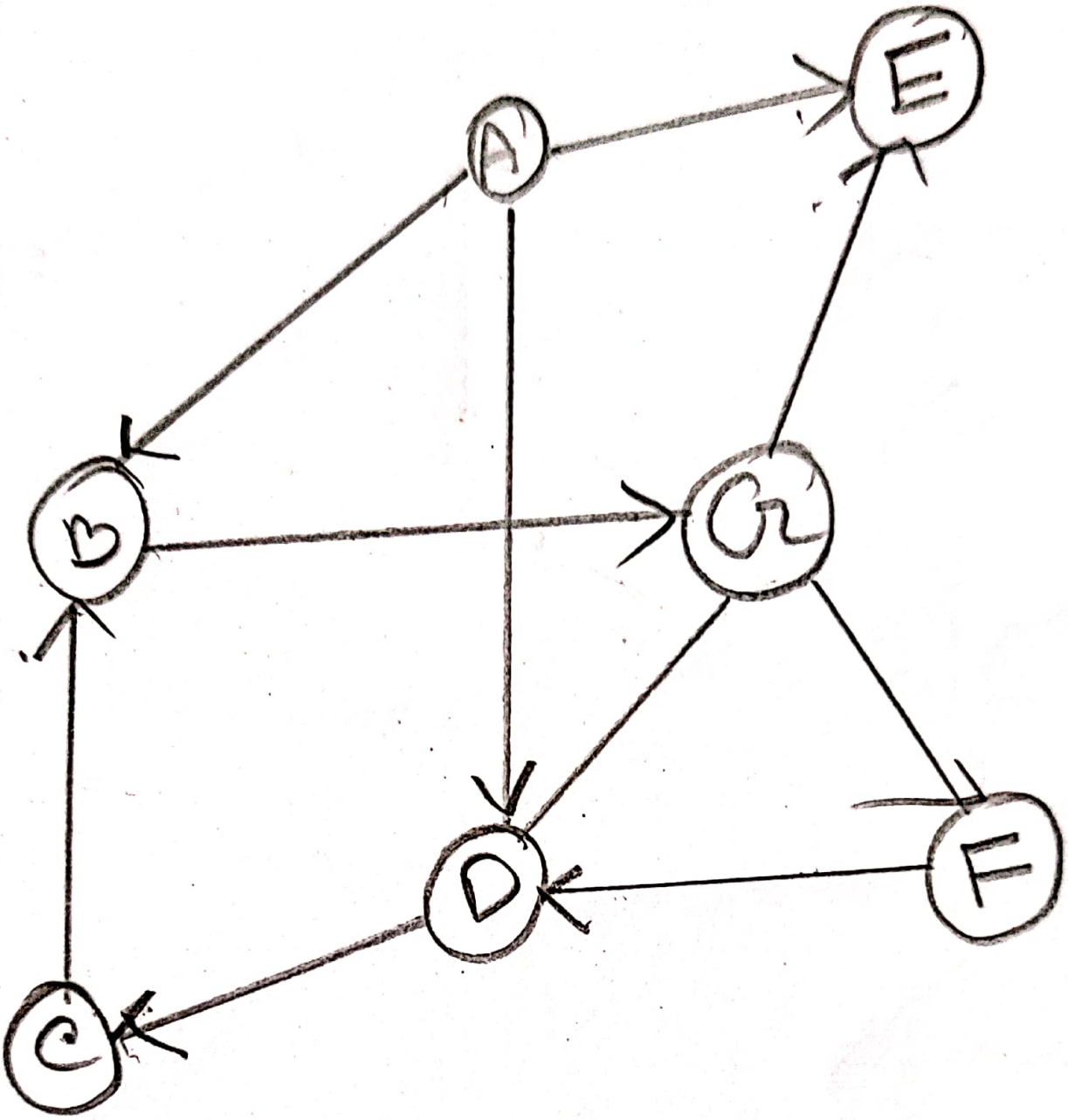
```
Height : 3
Level  : 3
Level  : 4
Pre-Order
10 20 40 50 30 70 80 90 100
IN-Order
40 20 50 10 70 30 90 80 100
post-Order
40 50 20 70 90 100 80 30 10
Not Same

Same
Copy of Binary Tree:
10 20 40 50 30 70 80 90 100
Main Binary Tree
10 20 40 50 30 70 80 90 100
```

In [ ]:

```

```


A
E
B
C
D
F