In [1]:
```
#take a linked list,add elements and reverse the first half of the linked list
```

In [4]:
```
"""
Double LL
i) inserting ele
ii) deleting ele
iii) traversing ele
iv) searching

Trees
i) binary trees
ii) binary search trees
iii) AVL trees
iv) B-trees
"""
#determine how many prime numbers are there in the linked list
```

Out[4]:
```
'\nDouble LL\ni) inserting ele\nii) deleting ele\niii) traversing ele\niv) searching\n\nTrees\ni) binary trees\nii) binary search trees\niii) AVL trees\niv) B-trees\n'
```

In [ ]:
```
# Binary Tree has atmost 2 children nodes
# Binary Search Tree ""   ""     ""    ""
# elements in left subtree should be less than root node
# and elements on right sub tree should be greater than root node

# Inorder -> left -root-right
# Preorder -> root-left-right
# Postorder -> left-right-root
# Levelorder -> top-bottom -> level0-level1-level2-....-leveln
# for level order travesal, we use queues
```

In [1]:
```
class Node:
    def __init__(self,value):
        self.data=value
        self.left=None
        self.right=None

class BSTree:
    def add_ele(self,root,value):

        new_node=Node(value) #create a new node to add an ele
        if new_node.data < root.data:
            if root.left!=None:
                self.add_ele(root.left,value)
            else:
                root.left=new_node
        else:
            if root.right!=None:
                self.add_ele(root.right,value)
            else:
                root.right=new_node



    def inorder(self,root):
        if root.left!=None:
            self.inorder(root.left)
        print(root.data)
        if root.right!=None:
            self.inorder(root.right)


    def preorder(self,root):
        print(root.data)
        if root.left!=None:
            self.preorder(root.left)
        if root.right!=None:
            self.preorder(root.right)

    def postorder(self,root):
        if root.left!=None:
            self.postorder(root.left)
        if root.right!=None:
            self.postorder(root.right)
        print(root.data)

    def levelorder(self,root):
        q=[]
        q.append(root)
        while len(q)!=0:
            ele=q.pop(0)
            print(ele.data,end=", ")
            if ele.left:
                q.append(ele.left)
            if ele.right:
                q.append(ele.right)


ob=BSTree()
root=Node(10)
ob.add_ele(root,7)
ob.add_ele(root,40)
ob.add_ele(root,5)
ob.add_ele(root,9)
ob.add_ele(root,15)
ob.add_ele(root,60)
print("INORDER : ")
ob.inorder(root)
print("inorder completed")
print()
print("PREORDER : ")
ob.preorder(root)
print("preorder completed")
print()
print("POSTORDER : ")
ob.postorder(root)
print("Postorder completed")
ob.levelorder(root)
```

```
INORDER :
5
7
9
10
15
40
60
inorder completed

PREORDER :
10
7
5
9
40
15
60
preorder completed

POSTORDER :
5
9
7
15
60
40
10
Postorder completed
10, 7, 40, 5, 9, 15, 60,
```

In [ ]: