Binary Search Tree

1] Search in BST recursive :

```
def searchBst(root,key):
    if root is None:
        return False

    elif root.key==key:
        return True

    elif root.left>key:
        return searchBst(root.left,key)
    else:
        return searchBst(root.right,key)
```

2] Search in BST iterative :

```
def searchBst(root,key):
    while root is not None:

        if root.key==key:
            return True
        elif root.left>key:
            root=root.left
        else:
            root=root.right
    return False
```

3] BST insert recursive solution :

```
class Node:
    def __init__(self, key):
        self.left = None
        self.key = key
        self.right = None

def insert(root,key):
    if root==None:
        return Node(key)
    elif root.key==key:
        return root
    elif root.left>key:
        root.left=insert(root.left,key)
    else:
        root.right=insert(root.right,key)
    return root
```

4] BST insert Iterative solution :

```
class Node:
    def __init__(self,key):
        self.left=None
        self.right=None
        self.key=key

def insert(root,key):
    parent=None
    curr=root

    while curr!=None:
        parent=curr

        if curr.key==key:
            return root

        elif curr.key<key:
            curr=curr.left

        else:
            curr=curr.right

    if parent==None:
        return Node(key)

    if parent.key>key:
        parent.left=Node(key)

    else:
        parent.right=Node(key)

    return root
```

5] BST delete in Python :

```
class Node:
    def __init__(self,key):
        self.left=None
        self.key=key
        self.right=None


def getSucc(curr,key):
    while curr.left!=None:
        curr=curr.left

    return curr.key

def deleteNode(root,key):
    if root==None:
        return

    if root.key>key:
        root.left=deleteNode(root.left,key)

    if root.key<key:
        root.right=deleteNode(root.right,key)

    else:
        if root.left==None:
            return root.right
        elif root.right==None:
            return root.left

        else:
            succ=getSucc(root.right,key)
            root.key=succ
            root.right=deleteNode(root.right,succ)

    return root
```

6] BST floor in python :

```
#floor mean the vlaue which is closer to less than or equal
class Node:
    def __init__(self,key):
        self.left=None
        self.key=key
        self.right=None


def getFloor(root,x):
    res=None

    while root !=None:
        if root.key==x:
            return root

        elif root.key>x:
            root=root.left

        else:
            res=root
            root=root.right

    return res
```

7] Ceiling in BST in python :

```
#ceiling means closer to greater or equal value
class Node:
    def __init__(self,key):
        self.left=None
        self.key=key
        self.right=None


def getCeil(root,x):
    res=None
    while root!=None:
        if root.key==x:
            return root

        elif root.key<x:
            root=root.right

        else:
            res=root
            root=root.left

    return res
```