

```

In [10]: 1 class Node:
2     def __init__(self,value):
3         self.value=value
4         self.left=None
5         self.right=None
6 class BST:
7     def __init__(self):
8         self.root=None
9     def Insert(self,value):
10        self.root=self.__Insert(self.root,value)
11    def __Insert(self,root,value):
12        if root==None:
13            root=Node(value)
14        else:
15            if value<root.value:
16                if root.left == None:
17                    root.left=Node(value)
18                else:
19                    root.left=self.__Insert(root.left,value)
20            else:
21                if root.right == None:
22                    root.right=Node(value)
23                else:
24                    root.right=self.__Insert(root.right,value)
25        return root
26    def InOrder(self):
27        return self.__inOrder(self.root)
28    def __inOrder(self,root):
29        if root:
30            self.__inOrder(root.left)
31            print(root.value)
32            self.__inOrder(root.right)
33    def PostOrder(self):
34        return self.__postOrder(self.root)
35    def __postOrder(self,root):
36        if root:
37            self.__postOrder(root.left)
38            self.__postOrder(root.right)
39            print(root.value)
40    def PreOrder(self):
41        return self.__preOrder(self.root)
42    def __preOrder(self,root):
43        if root:
44            print(root.value)
45            self.__preOrder(root.left)
46            self.__preOrder(root.right)
47    def Height(self):
48        return self.__Height(self.root)
49    def __Height(self,node):
50        if node==None:
51            return 0
52        else:
53            leftH=self.__Height(node.left)
54            rightH=self.__Height(node.right)
55            if leftH>rightH:
56                return leftH+1
57            else:
58                return rightH+1
59    def FindMax(self):
60        return self.__Max(self.root).value
61    def __Max(self,node):
62        while node is not None:
63            if node.right is None:
64                break
65            node = node.right
66        return node
67    def FindMin(self):
68        return self.__Min(self.root).value
69    def __Min(self,node):
70        while node is not None:
71            if node.left is None:
72                break
73            node = node.left
74        return node
75    def Successor(self):
76        if self.root.right is None:
77            return None
78        else:
79            return self.__Min(self.root.right).value
80    def Predecessor(self):
81        if self.root.left is None:
82            return None
83        else:
84            return self.__Max(self.root.left).value
85    def Delete(self,value):
86        return self.__Delete(self.root,value)
87    def __Delete(self,root,value):
88        if root is None:
89            return root

```

```

90     if value < root.value:
91         root.left = self.__Delete(root.left,value)
92     elif value > root.value:
93         root.right = self.__Delete(root.right,value)
94     else:
95         if root.left is None:
96             temp = root.right
97             root = None
98             return temp
99         elif root.right is None:
100             temp = root.left
101             root = None
102             return temp
103         temp = self.__FindMin(root.right)
104         root.value = temp.value
105         root.right = self.__Delete(root.right,temp.value)
106     return root
107 def Search(self,value):
108     return self.__Search(self.root,value)
109 def __Search(self,root,value):
110     if root is None or root.value == value:
111         return root
112     if root.value < value:
113         return self.__Search(root.right,value)
114     return self.__Search(root.left,value)
115 a=BST()
116 a.Insert(1)
117 a.Insert(3)
118 a.Insert(2)
119 a.Insert(40)
120 a.Insert(50)
121 a.Insert(0.5)
122 print("Height is ",a.Height())
123 print("Max is ",a.FindMax())
124 print("Min is ",a.FindMin())
125 print("Sucessor is",a.Successor())
126 print("Predecessor is",a.Predecessor())
127 print("-----InOrder-----")
128 a.InOrder()
129 print("-----PostOrder-----")
130 a.PostOrder()
131 print("-----PreOrder-----")
132 a.PreOrder()
133 a.Delete(50)
134 print("-----InOrder after deleting 50-----")
135 a.InOrder()
136 a.Search(40)

```

```

Height is 4
Max is 50
Min is 0.5
Sucessor is 2
Predecessor is 0.5
-----InOrder-----
0.5
1
2
3
40
50
-----PostOrder-----
0.5
2
50
40
3
1
-----PreOrder-----
1
0.5
3
2
40
50
-----InOrder after deleting 50-----
0.5
1
2
3
40

```

Out[10]: 40