```
using System;
namespace Tree_1 {
    class Program {
        class Node {
            public int data { get; set; }
            public Node Left { get; set; }
            public Node Right { get; set; }
        }
        class BinaryTree {
            public Node Root { get; set; }
            public Node tmp_node { get; set; }
            public void Insert(int Value) {
                Node Parent = null;
                Node tmp = this.Root;
                while (tmp != null) {
                    Parent = tmp;
                    if (Value < tmp.data) {
                        tmp = tmp.Left;
                    } else if (Value > tmp.data) {
                        tmp = tmp.Right;
                    } else return;
                }
                Node newNode = new Node();
                newNode.data = Value;
                newNode.Right = null;
                newNode.Left = null;
                if (this.Root == null) {
                    this.Root = newNode;
                } else {
                    if (newNode.data > Parent.data) Parent.Right = newNode;
                    else Parent.Left = newNode;
                }
                return;
            }
            public void PrintTree(Node Parent, String indent, bool last) {
                Console.WriteLine(indent + "+- " + Parent.data);
                indent += last ? "   " : "|  ";
                if (Parent.Left != null && Parent.Right != null)
                {
                    PrintTree(Parent.Left, indent, false);
                    PrintTree(Parent.Right, indent, true);
                }
                else if (Parent.Left != null) { PrintTree(Parent.Left, indent, true); }
                else if (Parent.Right != null) { PrintTree(Parent.Right, indent, true); }
            }
            public void Search(int Value) {
                Node Parent = null;
                Node tmp = this.Root;
                while (tmp != null) {
                    Parent = tmp;
                    if (Parent.data.Equals(Value)) {
                        tmp_node = Parent;
                        return;
                    } else if (Value < tmp.data) {
                        tmp = tmp.Left;
                    } else if (Value > tmp.data) {
                        tmp = tmp.Right;
                    } else {
                        return;
```

```
                    }
                }
            }
            public Node Remove(int Value)
            {
                Node parent = null;
                Node pointer = this.Root;
                while (pointer != null && pointer.data != Value) {
                    parent = pointer;
                    pointer = (Value < pointer.data) ? pointer.Left : pointer.Right;
                }
                if (pointer == null) return this.Root;
                if (pointer.Left == null && pointer.Right == null) {
                    if (pointer != this.Root) {
                        if (parent.Left == pointer) {
                            parent.Left = null;
                        } else {
                            parent.Right = null;
                        }
                    } else {
                        this.Root = null;
                    }
                } else if (pointer.Left != null && pointer.Right != null) {
                    Node successor = Bottom_Leaf(pointer.Right);
                    // remove data from buttom leaves that are move to node
                    Remove(successor.data);
                    pointer.data = successor.data;
                } else {
                    Node child = (pointer.Left != null) ? pointer.Left: pointer.Right;
                    if (pointer != this.Root) {
                        if (pointer == parent.Left){
                            parent.Left = child;
                        } else {
                            parent.Right = child;
                        }
                    } else {
                        this.Root = child;
                    }
                }
                return this.Root;
            }
            public Node Bottom_Leaf(Node pointer) {
                while (pointer.Left != null) {
                    pointer = pointer.Left;
                }
                return pointer;
            }
        }
        public static void Main(string[] args) {
            BinaryTree BT = new BinaryTree();
            int[] keys = { 50, 20, 70, 30, 110, 10, 15, 60, 90, 150 };
            foreach (int i in keys){
                BT.Insert(i);
            }
            BT.PrintTree(BT.Root, "Leaf     ", true);
            BT.Remove(10);
            BT.PrintTree(BT.Root, "After    ", true);
        }
    }
}
```