

```

1 using System;
2 namespace Tree_1 {
3     class Program {
4         class Node {
5             public int data { get; set; }
6             public Node Left { get; set; }
7             public Node Right { get; set; }
8         }
9         class BinaryTree {
10             public Node Root { get; set; }
11             public Node tmp_node { get; set; }
12             public void Insert(int Value) {
13                 Node Parent = null;
14                 Node tmp = this.Root;
15                 while (tmp != null) {
16                     Parent = tmp;
17                     if (Value < tmp.data) {
18                         tmp = tmp.Left;
19                     } else if (Value > tmp.data) {
20                         tmp = tmp.Right;
21                     } else return;
22                 }
23                 Node newNode = new Node();
24                 newNode.data = Value;
25                 newNode.Right = null;
26                 newNode.Left = null;
27                 if (this.Root == null) {
28                     this.Root = newNode;
29                 } else {
30                     if (newNode.data > Parent.data) Parent.Right = newNode;
31                     else Parent.Left = newNode;
32                 }
33                 return;
34             }
35             public void PrintTree(Node Parent, String indent, bool last) {
36                 Console.WriteLine(indent + "+- " + Parent.data);
37                 indent += last ? " " : "| ";
38                 if (Parent.Left != null && Parent.Right != null)
39                 {
40                     PrintTree(Parent.Left, indent, false);
41                     PrintTree(Parent.Right, indent, true);
42                 }
43                 else if (Parent.Left != null) { PrintTree(Parent.Left, indent, true); }
44                 else if (Parent.Right != null) { PrintTree(Parent.Right, indent, true); }
45             }
46             public void Search(int Value) {
47                 Node Parent = null;
48                 Node tmp = this.Root;
49                 while (tmp != null) {
50                     Parent = tmp;
51                     if (Parent.data.Equals(Value)) {
52                         tmp_node = Parent;
53                         return;
54                     } else if (Value < tmp.data) {
55                         tmp = tmp.Left;
56                     } else if (Value > tmp.data) {
57                         tmp = tmp.Right;
58                     } else {
59                         return;

```

```

60         }
61     }
62 }
63 public Node Remove(int Value)
64 {
65     Node parent = null;
66     Node pointer = this.Root;
67     while (pointer != null && pointer.data != Value) {
68         parent = pointer;
69         pointer = (Value < pointer.data) ? pointer.Left : pointer.Right;
70     }
71     if (pointer == null) return this.Root;
72     if (pointer.Left == null && pointer.Right == null) {
73         if (pointer != this.Root) {
74             if (parent.Left == pointer) {
75                 parent.Left = null;
76             } else {
77                 parent.Right = null;
78             }
79         } else {
80             this.Root = null;
81         }
82     } else if (pointer.Left != null && pointer.Right != null) {
83         Node successor = Bottom_Leaf(pointer.Right);
84         // remove data from bottom leaves that are move to node
85         Remove(successor.data);
86         pointer.data = successor.data;
87     } else {
88         Node child = (pointer.Left != null) ? pointer.Left : pointer.Right;
89         if (pointer != this.Root) {
90             if (pointer == parent.Left){
91                 parent.Left = child;
92             } else {
93                 parent.Right = child;
94             }
95         } else {
96             this.Root = child;
97         }
98     }
99     return this.Root;
100 }
101 public Node Bottom_Leaf(Node pointer) {
102     while (pointer.Left != null) {
103         pointer = pointer.Left;
104     }
105     return pointer;
106 }
107 }
108 public static void Main(string[] args) {
109     BinaryTree BT = new BinaryTree();
110     int[] keys = { 50, 20, 70, 30, 110, 10, 15, 60, 90, 150 };
111     foreach (int i in keys){
112         BT.Insert(i);
113     }
114     BT.PrintTree(BT.Root, "Leaf", true);
115     BT.Remove(10);
116     BT.PrintTree(BT.Root, "After", true);
117 }
118 }
119 }

```