C# Binary Tree Code

```
using System;
using System.Collections.Generic;

class Node
{
    public int Value;
    public Node Left, Right;

    public Node(int value)
    {
        Value = value;
        Left = Right = null;
    }
}

class BinarySearchTree
{
    public Node Root;

    public void Insert(int value)
    {
        Root = InsertRec(Root, value);
    }

    private Node InsertRec(Node root, int value)
    {
        if (root == null)
            return new Node(value);

        if (value < root.Value)
            root.Left = InsertRec(root.Left, value);
        else
            root.Right = InsertRec(root.Right, value);

        return root;
    }

    public void Preorder(Node node)
    {
        if (node == null) return;
        Console.Write(node.Value + " ");
        Preorder(node.Left);
        Preorder(node.Right);
    }

    public void Inorder(Node node)
    {
        if (node == null) return;
        Inorder(node.Left);
        Console.Write(node.Value + " ");
        Inorder(node.Right);
    }

    public void Postorder(Node node)
    {
        if (node == null) return;
        Postorder(node.Left);
        Postorder(node.Right);
        Console.Write(node.Value + " ");
    }

    public void LevelOrder()
    {
        if (Root == null) return;

        Queue<Node> q = new Queue<Node>();
        q.Enqueue(Root);

        while (q.Count > 0)
        {
            Node temp = q.Dequeue();
            Console.Write(temp.Value + " ");

            if (temp.Left != null) q.Enqueue(temp.Left);
            if (temp.Right != null) q.Enqueue(temp.Right);
        }
    }
}

class Program
{
    static void Main()
    {
        BinarySearchTree bst = new BinarySearchTree();

        Console.Write("How many numbers to insert? ");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter the numbers:");

        for (int i = 0; i < n; i++)
        {
            int value = int.Parse(Console.ReadLine());
            bst.Insert(value);
        }

        Console.Write("Preorder: ");
        bst.Preorder(bst.Root);
        Console.WriteLine();

        Console.Write("Inorder: ");
        bst.Inorder(bst.Root);
        Console.WriteLine();

        Console.Write("Postorder: ");
        bst.Postorder(bst.Root);
        Console.WriteLine();

        Console.Write("Level order: ");
        bst.LevelOrder();
        Console.WriteLine();
    }
}
```