```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  namespace BinarySearchTree
5  {
6      class Program
7      {
8          static Random random;
9          static void Main(string[] args)
10         {
11             List<string> nameList = new List<string>();
12             BinarySearchTree strTree = new BinarySearchTree();
13             int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
14             random = new Random(seed);
15
16             int n = 15;
17             for (int i = 0; i < n; i++)
18             {
19                 string s = RandomName(10);
20                 nameList.Add(s);
21                 strTree.Insert(s);
22             }
23             nameList.Add(RandomName(10));
24             Console.WriteLine(" Binary Search Tree \n");
25             strTree.Print();
26             Console.WriteLine("\n Search Test \n");
27             foreach (var s in nameList)
28             {
29                 Console.Write(strTree.Contains(s).ToString() + " ");
30             }
31             Console.WriteLine("\n");
32         }
33         static string RandomName(int size)
34         {
35             StringBuilder builder = new StringBuilder();
36             char ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 *
                 random.NextDouble() + 65)));
37             builder.Append(ch);
38             for (int i = 1; i < size; i++)
39             {
40                 ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 *
                     random.NextDouble() + 97)));
41                 builder.Append(ch);
42             }
43             return builder.ToString();
44         }
45     }
46 }
47
```

```
1  using System;
2  namespace BinarySearchTree
3  {
4      class TreeNode
5      {
6          public string Element { get; set; }
7          public TreeNode Left { get; set; }
8          public TreeNode Right { get; set; }
9          public int ElementNum { get; set; }
10
11         public TreeNode(string element, int num)
12         {
13             this.Element = element;
14             this.ElementNum = num;
15         }
16     }
17     class BinarySearchTree
18     {
19         public TreeNode Root { get; set; }
20         int count;
21         public BinarySearchTree()
22         {
23             this.Root = null;
24             count = 0;
25         }
26         public void Insert(string x)
27         {
28             this.Root = Insert(x, this.Root);
29         }
30         public bool Contains(string x)
31         {
32             return Contains(x, this.Root);
33         }
34         public void Print()
35         {
36             Print(this.Root);
37         }
38         private bool Contains(string x, TreeNode t)
39         {
40             while (t != null)
41             {
42                 if ((x as IComparable).CompareTo(t.Element) < 0)
43                 {
44                     t = t.Left;
45                 }
46                 else if ((x as IComparable).CompareTo(t.Element) > 0)
47                 {
48                     t = t.Right;
49                 }
50                 else
51                 {
52                     return true;
```

```
53                  }
54              }
55              return false;
56          }
57          protected TreeNode Insert(string x, TreeNode t)
58          {
59              if (t == null)
60              {
61                  t = new TreeNode(x, count++);
62              }
63              else if ((x as IComparable).CompareTo(t.Element) < 0)
64              {
65                  t.Left = Insert(x, t.Left);
66              }
67              else if ((x as IComparable).CompareTo(t.Element) > 0)
68              {
69                  t.Right = Insert(x, t.Right);
70              }
71              else
72              {
73                  // throw new Exception("Duplicate item");
74              }
75
76              return t;
77          }
78          private void Print(TreeNode t)
79          {
80              if (t == null)
81              {
82                  return;
83              }
84              else
85              {
86                  Print(t.Left);
87                  if (t.Left != null) Console.Write("{0,3:N0} <<- ",
                      t.Left.ElementNum); else Console.Write("        ");
88                  Console.Write("{0,3:N0} {1} ", t.ElementNum, t.Element);
89                  if (t.Right != null) Console.WriteLine(" ->> {0,3:N0}",
                      t.Right.ElementNum); else Console.WriteLine("        ");
90                  Print(t.Right);
91              }
92          }
93      }
94  }
95
```