```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using System.IO;
5
6  namespace BinarySearchTree
7  {
8      class Program
9      {
10         static Random random;
11         static void Main(string[] args)
12         {
13             List<string> nameList = new List<string>();
14             BinarySearchTree strTree = new BinarySearchTree();
15             int seed = (int)DateTime.Now.Ticks & 0x0000FFFF;
16             random = new Random(seed);
17
18             int n = 15;
19             int k = 10;
20
21             for (int i = 0; i < n; i++)
22             {
23                 string s = RandomName(k);
24                 nameList.Add(s);
25                 strTree.Insert(s);
26             }
27             nameList.Add(RandomName(k));
28
29             Console.WriteLine(" Binary Search Tree \n");
30             strTree.Print();
31
32             //Console.WriteLine("\n Search Test \n");
33             //foreach (var s in nameList)
34             //{
35             //    Console.Write(strTree.Contains(s).ToString() + " ");
36             //}
37             //Console.WriteLine("\n");
38
39             Console.WriteLine("\n Search FILE Test \n");
40             string filename = @"mydatatree.dat";
41             strTree.WriteToFile(filename, n);
42
43             using (FileStream fs = new FileStream(filename, FileMode.Open,
                 FileAccess.ReadWrite))
44             {
45                 foreach (var s in nameList)
46                 {
47                     Console.Write(strTree.FileContains(fs, s).ToString() + " ");
48                 }
49                 Console.WriteLine("\n");
50             }
51         }
```

```
52          static string RandomName(int size)
53          {
54              StringBuilder builder = new StringBuilder();
55              char ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 *
                  random.NextDouble() + 65)));
56              builder.Append(ch);
57              for (int i = 1; i < size; i++)
58              {
59                  ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 *
                      random.NextDouble() + 97)));
60                  builder.Append(ch);
61              }
62              return builder.ToString();
63          }
64      }
65
66  }
67
```

```cs
using System;
using System.IO;
using System.Text;

namespace BinarySearchTree
{
    class TreeNode
    {
        public string Element { get; set; }
        public TreeNode Left { get; set; }
        public TreeNode Right { get; set; }
        public int ElementNum { get; set; }

        public TreeNode(string element, int num)
        {
            this.Element = element;
            this.ElementNum = num;
        }
    }

    class BinarySearchTree
    {
        public TreeNode Root { get; set; }
        int count;

        public BinarySearchTree()
        {
            this.Root = null;
            count = 0;
        }

        public void Insert(string x)
        {
            this.Root = Insert(x, this.Root);
        }

        public bool Contains(string x)
        {
            return Contains(x, this.Root);
        }

        public bool FileContains(FileStream fs, string x)
        {
            return FileContains(fs, x, 0, this.Root.Element.Length);
        }
        public void Print()
        {
            Print(this.Root);
        }
        public void WriteToFile(string filename, int n)
        {
            byte[][] bufTree = new byte[n][];
```

```
53
54              BuildBufTree(bufTree, this.Root, this.Root.Element.Length);
55
56              if (File.Exists(filename)) File.Delete(filename);
57              try
58              {
59                  using (BinaryWriter writer = new BinaryWriter(File.Open(filename,
                       FileMode.Create)))
60                  {
61                      foreach (var item in bufTree)
62                      {
63                          for (int j = 0; j < item.Length; j++)
64                              writer.Write(item[j]);
65                      }
66                  }
67              }
68              catch (IOException ex)
69              {
70                  Console.WriteLine(ex.ToString());
71              }
72          }
73          private void BuildBufTree(byte[][] bufTree, TreeNode t, int k)
74          {
75              int mn = -1;
76              if (t == null)
77              {
78                  return;
79              }
80              else
81              {
82                  BuildBufTree(bufTree, t.Left, k);
83
84                  int i = t.ElementNum;
85                  bufTree[i] = new byte[k + 8];
86
87                  if (t.Left != null)
88                      BitConverter.GetBytes(t.Left.ElementNum).CopyTo(bufTree[i],
                           0);
89                  else
90                      BitConverter.GetBytes(mn).CopyTo(bufTree[i], 0);
91                  Encoding.ASCII.GetBytes(t.Element).CopyTo(bufTree[i], 4);
92                  if (t.Right != null)
93                      BitConverter.GetBytes(t.Right.ElementNum).CopyTo(bufTree[i],
                           k + 4);
94                  else
95                      BitConverter.GetBytes(mn).CopyTo(bufTree[i], k + 4);
96
97                  BuildBufTree(bufTree, t.Right, k);
98              }
99          }
100         private bool Contains(string x, TreeNode t)
101         {
```

```
102             while (t != null)
103             {
104                 if ((x as IComparable).CompareTo(t.Element) < 0)
105                 {
106                     t = t.Left;
107                 }
108                 else if ((x as IComparable).CompareTo(t.Element) > 0)
109                 {
110                     t = t.Right;
111                 }
112                 else
113                 {
114                     return true;
115                 }
116             }
117             return false;
118         }
119         private bool FileContains(FileStream fs, string x, int t, int k)
120         {
121             int ak = k + 8;
122             byte[] data = new byte[ak];
123             while (t >= 0)
124             {
125                 fs.Seek(t * ak, SeekOrigin.Begin);
126                 fs.Read(data, 0, ak);
127                 int tLeft = BitConverter.ToInt32(data, 0);
128                 string element = Encoding.ASCII.GetString(data, 4, k);
129                 int tRight = BitConverter.ToInt32(data, k + 4);
130 
131                 if ((x as IComparable).CompareTo(element) < 0)
132                 {
133                     t = tLeft;
134                 }
135                 else if ((x as IComparable).CompareTo(element) > 0)
136                 {
137                     t = tRight;
138                 }
139                 else
140                 {
141                     return true;
142                 }
143             }
144             return false;
145         }
146 
147         protected TreeNode Insert(string x, TreeNode t)
148         {
149             if (t == null)
150             {
151                 t = new TreeNode(x, count++);
152             }
153             else if ((x as IComparable).CompareTo(t.Element) < 0)
```

```
154             {
155                 t.Left = Insert(x, t.Left);
156             }
157             else if ((x as IComparable).CompareTo(t.Element) > 0)
158             {
159                 t.Right = Insert(x, t.Right);
160             }
161             else
162             {
163                 // throw new Exception("Duplicate item");
164             }
165 
166             return t;
167         }
168 
169         private void Print(TreeNode t)
170         {
171             if (t == null)
172             {
173                 return;
174             }
175             else
176             {
177                 Print(t.Left);
178                 if (t.Left != null) Console.Write("{0,3:N0} <<- ",
                      t.Left.ElementNum); else Console.Write("        ");
179                 Console.Write("{0,3:N0} {1} ", t.ElementNum, t.Element);
180                 if (t.Right != null) Console.WriteLine(" ->> {0,3:N0}",
                      t.Right.ElementNum); else Console.WriteLine("        ");
181                 Print(t.Right);
182             }
183         }
184     }
185 }
```