```
using System;
//using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Unit4.CollectionsLib;
using Unit4.BinTreeCanvasLib;
using Unit4.BinTreeUtilsLib;

namespace BinNodeProj
{
    class BTmethodsCS
    {
        public static void main(String[] args)
        {
            int MaxNodes = 10;       //  מספר הצמתים בעץ
            int low = -4;            //  ערך מינימלי
            int high = 8;            //  ערך מקסימאלי

            //--- MaxNodes הפעולה מחזירה עץ בינארי אקראי, שמספר צמתיו הוא ---
            //---        high -ו low כל ערכיו הם מספרים שלמים אקראיים בין ---
            BinNode<int> bt = BinTreeUtils.BuildRandomTree(MaxNodes, low, high);

            //--- הפעולה מציגה תאור גראפי של העץ ---
            TreeCanvas.AddTree(bt);

            //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            Console.WriteLine(NumNodes(bt) + " : מספר הצמתים בעץ ");
            Console.WriteLine(SumNodes(bt) + " : סכום הצמתים בעץ ");
            Console.WriteLine();

            Console.WriteLine(NumPositive(bt) + " : מספר הצמתים החיוביים בעץ ");
            Console.WriteLine(SumPositive(bt) + " : סכום הצמתים החיוביים בעץ ");
            Console.WriteLine();

            Console.WriteLine(CountLeftSons(bt) + " : מספר הבנים השמאליים");
            Console.WriteLine(SumLeftSons(bt) + " : סכום הבנים השמאליים");
            Console.WriteLine();

            Console.WriteLine(CountRightSons(bt) + " : מספר הבנים הימניים");
            Console.WriteLine(SumRightSons(bt) + " : סכום הבנים הימניים");
            Console.WriteLine();

            Console.WriteLine(CountLeaves(bt) + " : מספר העלים");
            Console.WriteLine(SumLeaves(bt) + " : סכום העלים");

            Console.WriteLine(MaxInTree(bt) + " : ערך מקסימאלי בעץ");
            Console.WriteLine(Height(bt) + " : גובה העץ");

            Console.WriteLine(IsBalancedTree(bt) + " : עץ מאוזן");
        }

        /*****************************************
         *      אוסף פעולות שימושיות בעץ בינארי     *
         *****************************************/

        //--- מספר הצמתים בעץ ---
        public static int NumNodes(BinNode<int> bt)
        {
            if (bt == null)
                return 0;
            return 1 +                          //-- בגלל הצומת הנוכחי
                    NumNodes(bt.GetLeft()) +    //-- מספר הצמתים בבן השמאלי
                    NumNodes(bt.GetRight());    //-- מספר הצמתים בבן הימני
        }

        //--- סכום הצמתים בעץ ---
        public static int SumNodes(BinNode<int> bt)
        {
            if (bt == null)
                return 0;
            return bt.GetValue() +              //  ערך הצומת הנוכחי
                    SumNodes(bt.GetLeft()) +    //  סכום הצמתים בבן השמאלי
                    SumNodes(bt.GetRight());    //  סכום הצמתים בבן הימני
        }
```

```
77
78          //--- *** מספר הצמתים המקיימים תנאי **** ---
79          //  --- מספר הצמתים החיוביים בעץ ---
80          public static int NumPositive(BinNode<int> bt)
81          {
82              if (bt == null)
83                  return 0;               //---          זהו התנאי המבוקש
84              if (bt.GetValue() > 0)      //---  אפשר להחליף אותו בתנאי אחר
85                  return 1 + NumPositive(bt.GetLeft()) +
86                             NumPositive(bt.GetRight());
87              return NumPositive(bt.GetLeft()) +
88                     NumPositive(bt.GetRight());
89          }
90
91          //  --- סכום הצמתים החיוביים בעץ ---
92          public static int SumPositive(BinNode<int> bt)
93          {
94              if (bt == null)
95                  return 0;
96              if (bt.GetValue() > 0)
97                  return bt.GetValue() +
98                         SumPositive(bt.GetLeft()) +
99                         SumPositive(bt.GetRight());
100             return SumPositive(bt.GetLeft()) +
101                    SumPositive(bt.GetRight());
102         }
103
104         //  --- מספר הבנים השמאליים בעץ ---
105         public static int CountLeftSons(BinNode<int> bt)
106         {
107             if (bt == null)
108                 return 0;
109             if (bt.HasLeft())
110                 return 1 + CountLeftSons(bt.GetLeft()) +
111                            CountLeftSons(bt.GetRight());
112             return CountLeftSons(bt.GetRight());
113         }
114
115         //  --- מספר הבנים הימניים בעץ ---
116         public static int CountRightSons(BinNode<int> bt)
117         {
118             if (bt == null)
119                 return 0;
120             if (bt.HasRight())
121                 return 1 + CountRightSons(bt.GetLeft()) +
122                            CountRightSons(bt.GetRight());
123             return CountRightSons(bt.GetLeft());
124         }
125
126         //  --- סכום הבנים השמאליים בעץ ---
127         public static int SumLeftSons(BinNode<int> bt)
128         {
129             if (bt == null)
130                 return 0;
131             if (bt.HasLeft())
132                 return bt.GetLeft().GetValue() +    // ערך הבן השמאלי
133                        SumLeftSons(bt.GetLeft()) +
134                        SumLeftSons(bt.GetRight());
135             return SumLeftSons(bt.GetRight());
136         }
137
138         //  --- סכום הבנים הימניים בעץ ---
139         public static int SumRightSons(BinNode<int> bt)
140         {
141             if (bt == null)
142                 return 0;
143             if (bt.HasRight())
144                 return bt.GetRight().GetValue() +
145                        SumRightSons(bt.GetLeft()) +
146                        SumRightSons(bt.GetRight());
147             return SumRightSons(bt.GetLeft());
148         }
149
150
151
152
```

```
153         //---        ?האם עלה      ---
154         //-- null אינו bt : הנחה
155         public static bool IsLeaf(BinNode<int> bt)
156         {
157             if (!bt.HasLeft() && !bt.HasRight())
158                 return true;
159             return false;
160 
161             /*
162              return (bt.GetLeft() ==  bt.GetRight()) ;      //  :ואפשר גם
163                                 // null כי בנים יכולים להיות שווים רק כאשר הם
164              */
165         }
166 
167         //--- מספר העלים בעץ ---
168         public static int CountLeaves(BinNode<int> bt)
169         {
170             if (bt == null)
171                 return 0;
172             if (IsLeaf(bt))
173                 return 1;
174             return CountLeaves(bt.GetLeft()) + CountLeaves(bt.GetRight());
175         }
176 
177         //  --- סכום העלים בעץ ---
178         public static int SumLeaves(BinNode<int> bt)
179         {
180             if (bt == null)
181                 return 0;
182             if (IsLeaf(bt))
183                 return bt.GetValue();
184             return SumLeaves(bt.GetLeft()) + SumLeaves(bt.GetRight());
185         }
186 
187         //--- עץ בינארי לא ריק ---
188         //--- הערך המקסימאלי בעץ ---
189         public static int MaxInTree(BinNode<int> bt)
190         {
191             if (bt == null)
192                 return 0;
193             int maxLeft = MaxInTree(bt.GetLeft());
194             int maxRight = MaxInTree(bt.GetRight());
195 
196             return Math.Max(bt.GetValue(), Math.Max(maxLeft, maxRight));
197         }
198 
199 
200         //--- פעולה המחזירה את גובה העץ ---
201         public static int Height(BinNode<int> bt)
202         {
203             if (bt == null) return -1;
204             if (IsLeaf(bt)) return 0;
205             return 1 + Math.Max(Height(bt.GetLeft()), Height(bt.GetRight()));
206         }
207 
208         //---            ?האם עץ מאוזן          ---
209         //--- עץ מאוזן - עץ שבו לכל צומת, הפרש הגבהים
210         //---               1 של שני בניו אינו עולה על ---
211         public static bool IsBalancedTree(BinNode<int> bt)
212         {
213             if (bt == null || IsLeaf(bt))
214                 return true;
215             if (Math.Abs(Height(bt.GetLeft()) - Height(bt.GetRight())) > 1)
216                 return false;
217             return IsBalancedTree(bt.GetLeft()) &&
218                    IsBalancedTree(bt.GetRight());
219         }
220     }
221 
222 }
223 
```