```cpp
#include <iostream>
using  namespace  std  ;

///////////  Array tree structure definition  ///////////////

struct  Cell
{
    int  data   ;
    int  left   ;
    int  right  ;
};

typedef  struct Cell * ATree  ;

int  tree_root(ATree atree, int nbre)
{
    if(nbre <= 0)
        return  -1  ;

    bool*  root_array  =  new  bool[nbre]   ;

    for(int i = 0; i < nbre ; i++)
        root_array[i]  =  true  ;

    for(int i = 0; i < nbre ; i++)
    {
        if(atree[i].left != -1)
            root_array[atree[i].left]  =  false  ;

        if(atree[i].right != -1)
            root_array[atree[i].right]  =  false  ;
    }

    for(int i = 0; i < nbre ; i++)
        if(root_array[i])
            return  i  ;

    return  -1  ;
}

///////////  Tree structure definition  ///////////////

struct  node
{
    int           data   ;
    struct node * left   ;
    struct node * right  ;
};

typedef  struct node *  Tree  ;

///////////  Queue structure definition  ///////////////

struct  element
{
    Tree   node  ;
    struct element * next  ;
};

typedef  struct element * Queue  ;

Queue  create_queue()
{
    return  NULL  ;
}

bool  is_empty_queue(Queue que)
{
    if(que == NULL)
        return  true  ;
    else
        return  false ;
}

Queue  enqueue(Queue que, Tree node)
{
    Queue  elm  =  new  element  ;
```

```
78      elm->node   =  node   ;
79      elm->next   =  NULL  ;
80
81      if(is_empty_queue(que))
82      {
83          que  =  elm  ;
84          return  que  ;
85      }
86
87      Queue  ptr  =  que  ;
88
89      while(ptr->next != NULL)
90          ptr  =  ptr->next  ;
91
92      ptr->next  =  elm  ;
93
94      return  que  ;
95  }
96
97  Queue  dequeue(Queue que, Tree &node)
98  {
99      if(!is_empty_queue(que))
100     {
101         Queue  ptr  =  que  ;
102
103         node =  que->node ;
104         que  =  que->next  ;
105         delete  ptr        ;
106
107         return  que  ;
108     }
109     else
110     {
111         cout  <<  "Error, the queue is already empty."  <<  endl  ;
112         return  NULL  ;
113     }
114 }
115
116 Tree  create_tree(ATree atree, int root)
117 {
118     if(root == -1)
119         return  NULL  ;
120
121     Tree  ptr  =  new  node  ;
122
123     ptr->data   =  atree[root].data  ;
124     ptr->left   =  create_tree(atree, atree[root].left)   ;
125     ptr->right  =  create_tree(atree, atree[root].right)  ;
126
127     return  ptr  ;
128 }
129
130 void  postfix_display(Tree tree)
131 {
132     if(tree != NULL)
133     {
134         postfix_display(tree->left)    ;
135         postfix_display(tree->right)   ;
136         cout  <<  " "  <<  tree->data  ;
137     }
138
139     return  ;
140 }
141
142 void  prefix_display(Tree tree)
143 {
144     if(tree != NULL)
145     {
146         cout  <<  " "  <<  tree->data  ;
147         prefix_display(tree->left)     ;
148         prefix_display(tree->right)    ;
149     }
150
151     return  ;
152 }
153
```

```
154 void  infix_display(Tree tree)
155 {
156     if(tree != NULL)
157     {
158         infix_display(tree->left)      ;
159         cout  <<  " "  <<  tree->data  ;
160         infix_display(tree->right)     ;
161     }
162 
163     return  ;
164 }
165 
166 int  nodes_number(Tree tree)
167 {
168     if(tree == NULL)
169     {
170         return  0  ;
171     }
172     else
173     {
174         return  1 + nodes_number(tree->left) + nodes_number(tree->right)  ;
175     }
176 }
177 
178 int  nodes_sum(Tree tree)
179 {
180     if(tree == NULL)
181     {
182         return  0  ;
183     }
184     else
185     {
186         return  tree->data + nodes_sum(tree->left) + nodes_sum(tree->right)  ;
187     }
188 }
189 
190 int  leaves_number(Tree tree)
191 {
192     if(tree == NULL)
193         return  0  ;
194 
195     if((tree->left == NULL)&&(tree->right == NULL))
196     {
197         return  1  ;
198     }
199     else
200     {
201         return  leaves_number(tree->left) + leaves_number(tree->right)  ;
202     }
203 }
204 
205 int  tree_height(Tree tree)
206 {
207     if(tree == NULL)
208     {
209         return  0  ;
210     }
211     else
212     {
213         return  1 + max(tree_height(tree->left), tree_height(tree->right))  ;
214     }
215 }
216 
217 bool  balanced_tree(Tree tree)
218 {
219     if(tree == NULL)
220         return  true  ;
221 
222     if(abs(tree_height(tree->left) - tree_height(tree->right)) > 1)
223         return  false  ;
224     else
225         return  balanced_tree(tree->left) && balanced_tree(tree->right)  ;
226 }
227 
```

```
228 void  by_level_display(Tree tree)
229 {
230     Queue  que  =  create_queue()  ;
231 
232     if(tree == NULL)
233     {
234         cout  <<  "Tree is empty."  <<  endl  ;
235         return  ;
236     }
237 
238     que  =  enqueue(que, tree)  ;
239 
240     while(!is_empty_queue(que))
241     {
242         Tree  ptr  ;
243 
244         que  =  dequeue(que, ptr)  ;
245 
246          cout  <<  ptr->data  <<  " "  ;
247 
248          if(ptr->left != NULL)
249             que  =  enqueue(que, ptr->left)   ;
250 
251          if(ptr->right != NULL)
252             que  =  enqueue(que, ptr->right)  ;
253     }
254 
255     cout  <<  endl  ;
256 
257     return  ;
258 }
259 
260 int  main()
261 {
262     Cell  atree[]  =  { {23, -1, -1}, {2, 4, 5}, {3, 3, 0}, {5, -1, -1}, {7, - 1, -1},
                          {11, 9, -1}, {13, -1, 2}, {37, 8, 1}, {41, 6, -1}, {19, -1, - 1} } ;
263 
264     Tree  tree  =  create_tree(atree, tree_root(atree, 10))  ;
265 
266     postfix_display(tree)  ;  cout  <<  endl  ;
267     prefix_display(tree)   ;  cout  <<  endl  ;
268     infix_display(tree)    ;  cout  <<  endl  ;
269 
270     cout  <<  "The number of nodes on the tree is : "   <<  nodes_number(tree)   <<  endl  ;
271     cout  <<  "The sum of nodes on the tree is : "      <<  nodes_sum(tree)      <<  endl  ;
272     cout  <<  "The number of leaves on the tree is : "  <<  leaves_number(tree)  <<  endl  ;
273     cout  <<  "The height/depth of the tree is : "      <<  tree_height(tree)    <<  endl  ;
274 
275     if(balanced_tree(tree))
276         cout <<  "The tree is balanced."      <<  endl  ;
277     else
278         cout <<  "The tree is not balanced."  <<  endl  ;
279 
280     by_level_display(tree)  ;
281 
282 
283     return  0  ;
284 }
```