Consider that **BST<E>** is a class that represents a **generic binary search tree** and **Node<E>** is a nested class that represents a **node of that tree**, as shown below.

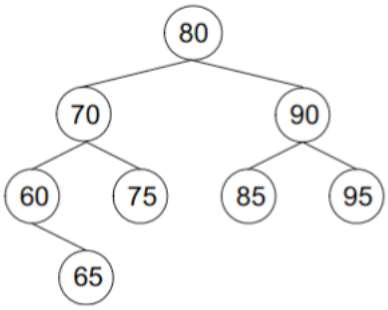
```
public class BST<E extends Comparable<E>>{
      protected Node<E> root = null;      // root of the tree
...
//-----------------------------------------------------------------
protected static class Node<E> {
  private E element;              // an element stored at this node
  private Node<E> left;          // a reference to the left child (if any)
  private Node<E> right;        // a reference to the right child (if any)
...
}
//-----------------------------------------------------------------
}
```

1. Implement the class TREE by inheriting from the BST, and add to it the following methods:

a) `public boolean contains(E element)` that verifies if an element is in the tree.

b) `public boolean isLeaf(E element)` that verifies if an element is in a leaf.

c) `public Iterable<E> ascdes()` - returns an iterable list with the elements of the left subtree in increasing order and the elements of the right subtree in descending order. According to the figure, the result is: 60, 65, 70, 75, 80, 95, 90, 85.

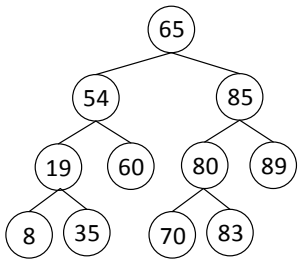


d) `public BST<E> autumnTree()` – returns a new binary search tree, identical to the original, but without the leaves.

e) `public int[] numNodesByLevel()` – returns an array with the number of nodes in each level of the tree, position 0 number of nodes at level 0, position 1 number of nodes at level 1, …

Complementary Exercises

1. Add to the TREE<E> class a method that traverses a binary tree following the orientation of a sequence of 0s and 1s, such that a zero descends in the left subtree and a 1 descends in the right subtree. The method receives a binary tree and a string of 0s and 1s, and returns the element of the tree that matchs with the last element of the string, or null in case of the string does not end in a valid node. For example, for the tree below and with a string 101 the method returns the element 83, with a string 110 returns null.



2. Add to the TREE <E> class a method that returns a list with the top-half elements of the tree sorted in descending order. For the tree below the method should return the list {89, 85, 80, 65, 60, 54, 19}.