class RBNode

```
public class RBNode {

    enum Color {RED, BLACK};

    private RBNode left, right, parent;
    private Color color;
    private int elt;
```

method `isLeaf()`

```
public boolean isLeaf() {
      return (left == null && right == null);
}
```

method `redChildOK()`

```
public boolean redChildOK() {

        if (isLeaf()) return true;

        boolean leftOK = true;
        boolean rightOK = true;

        if (color == Color.RED) {
                if (left != null && left.color == Color.RED)  return false;
                if (right != null && right.color == Color.RED)  return false;
        }
        if (left != null)
                leftOK = left.redChildOK();

        if (leftOK && right != null)
                rightOK = right.redChildOK();

        return (leftOK && rightOK);
}
```

method numBlkOK()

```
public int numBlkOK() {
        if (isLeaf()) {
                if (color == Color.BLACK) return 1;
                else return 0;
        }

        int leftNumBlk = 0, rightNumBlk = 0;

        if (left != null)
                leftNumBlk = left.numBlkOK();

        if (right != null)
                rightNumBlk = right.numBlkOK();

        if (leftNumBlk == -1 || rightNumBlk == -1) return -1;
        if (leftNumBlk != rightNumBlk) return -1;

        if (color == Color.BLACK)
                leftNumBlk++;

        return leftNumBlk;
}
```

method `isAncestorOf(RBNode node)`

```
public boolean isAncestorOf(RBNode node) {

        if (this == node) return true;

        if (node.parent == null) return false;

        return isAncestorOf(node.parent);
}
```