```
from binarytree import *

def balanceado(tree):
    if tree.root==None:
        return True

    contador1=0
    contador2=0

    if tree.root.leftnode!=None:
        contador1=balanceadoR(tree.root.leftnode)

    if tree.root.rightnode!=None:
        contador2= balanceadoR(tree.root.rightnode)

    if contador1==contador2 or contador1 ==contador2-1 or
contador2==contador1-1:
        return True
    return False

def balanceadoR(nodo):
    if nodo==None:
        return 0

    alturaIzq= balanceadoR(nodo.leftnode)
    alturaDerecha= balanceadoR(nodo.rightnode)

    if alturaIzq>alturaDerecha:
        return alturaIzq+1
    else:
        return alturaDerecha+1

def searchK(nodo, k):
    if nodo==None:
        return None

    if nodo.key==k:
        return nodo
```

```
    izquierda= searchK(nodo.leftnode, k)
    if izquierda!=None:
        return izquierda

    derecha= searchK(nodo.rightnode, k)
    if derecha!=None:
        return derecha

def subarbol(tree1, tree2):
    if tree1.root==None or tree2.root==None:
        return None

    llave= tree2.root.key
    comienzo= searchK(tree1.root, llave)

    if comienzo==None:
        return False

    return subarbolR(comienzo, tree2.root)

def subarbolR(nodo1, nodo2):
    if nodo1==None and nodo2==None:
        return True

    if nodo1.key!=nodo2.key or nodo1.value!=nodo2.value:
        return False

    seguir=subarbolR(nodo1.leftnode, nodo2.leftnode)
    if seguir==True:
        return subarbolR(nodo1.rightnode, nodo2.rightnode)
    return seguir

def check(tree):
    if tree.root==None:
        return True

    return checkR(tree.root, float('-inf'), float('inf'))
```

```
def checkR(nodo, minimo, maximo):
    if nodo==None:
        return True

    if nodo.key<=minimo or nodo.key>=maximo:
        return False

    return checkR(nodo.leftnode, minimo, nodo.key) and
checkR(nodo.rightnode, nodo.key, maximo)
```