Ejercicio 1

```python
#Ejercicio 1
def rotateLeft(tree,avlnode):
    """Realiza la rotacion a la izquierda"""
    if tree.root == avlnode:
        nodeA = avlnode
        nodeB = nodeA.rightnode

        tree.root = nodeB
        nodeA.rightnode = nodeB.leftnode
        nodeB.leftnode = nodeA
        nodeA.parent = nodeB
        nodeB.parent = None
    else:
        nodeA = avlnode
        nodeB = nodeA.rightnode
        parentNode = nodeA.parent

        if parentNode.leftnode == nodeA:
            parentNode.leftnode = nodeB
        else:
            parentNode.rightnode = nodeB
        nodeA.rightnode = nodeB.leftnode
        nodeB.leftnode = nodeA
        nodeA.parent = nodeB
        nodeB.parent = None

    return tree
```

```python
def rotateRight(tree,avlnode):
    "Realiza la rotacion a la derecha"
    if tree.root == avlnode:
        nodeA = avlnode
        nodeB = nodeA.leftnode

        tree.root = nodeB
        nodeA.leftnode = nodeB.rightnode
        nodeB.rightnode = nodeA
        nodeB.parent = None
        nodeA.parent = nodeB

    else:
        nodeA = avlnode
        nodeB = nodeA.leftnode
        parentNode = nodeA.parent

        if parentNode.leftnode == nodeA:
            parentNode.leftnode = nodeB
        elif parentNode == nodeB:
            parentNode.rightnode = nodeB
        nodeA.leftnode = nodeB.rightnode
        nodeB.rightnode = nodeA
        nodeB.parent = parentNode
        nodeA.parent = nodeB

    return tree
```

Ejercicio 2

```python
#Ejercicio 2
def calculateBalance(root):
    """calcula el balance factor de un arbol binario de busqueda"""
    if root == None:
        return None
    if isinstance(root,AVLTree):
        root = root.root

    calculateBalance(root.leftnode)
    calculateBalance(root.rightnode)

    leftvalue = heightTree(root.leftnode)
    rightvalue = heightTree(root.rightnode)

    diffvalue = (leftvalue - rightvalue)

    root.bf = diffvalue
```

Ejercicio 3

```python
#Ejercicio 3
def Wrapper_reBalance(tree,root):
    if root == None:
        return None

    if isinstance(root,AVLTree):
        root = root.root


    Wrapper_reBalance(tree,root.leftnode)
    Wrapper_reBalance(tree,root.rightnode)

    while True:
        calculateBalance(tree)
        if root.bf < -1:
            if root.rightnode.bf <= 0:
                rotateLeft(tree, root)
            else:
                rotateRight(root, root.rightnode)
                rotateLeft(tree, root)
        elif root.bf > 1:
            if root.leftnode.bf >= 0:
                rotateRight(tree,root)
            else:
                rotateLeft(tree, root.leftnode)
                rotateRight(tree, root)
        else:
            break

    return tree
def reBalance(tree):
    return Wrapper_reBalance(tree,tree.root)
```

Ejercicio 4

```python
#Ejercicio 4
def wrapperInsert(node,currentNode):
    if node.value < currentNode.value:
        if currentNode.leftnode == None:
            currentNode.leftnode = node
            node.parent = currentNode
        else:
            wrapperInsert(node, currentNode.leftnode)
    else:
        if currentNode.rightnode == None:
            currentNode.rightnode = node
            node.parent = currentNode
        else:
            wrapperInsert(node, currentNode.rightnode)


def insert(tree,value,key):
    newNode = AVLNode(key=key,value=value)
    if tree.root == None:
        tree.root = newNode
        return tree

    wrapperInsert(newNode,tree.root)
    reBalance(root)
```

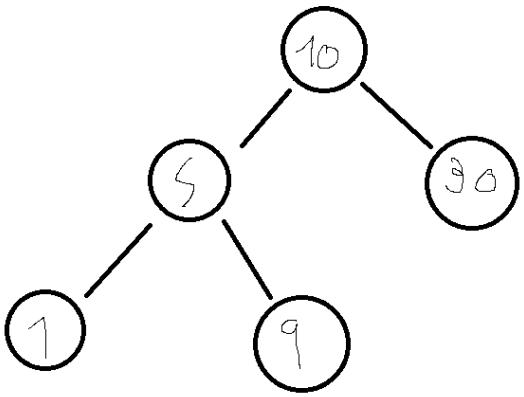
Ejercicio 5

```python
#Ejercicio 5
def findMinNode(B):
  if B.leftnode == None:
    return B
  return findMinNode(B.leftnode)


def findMaxNode(B):
  if B.rightnode == None:
    return B
  return findMaxNode(B.rightnode)

def deleteNode(currentNode,key):

  if currentNode == None:
    return

  if currentNode.key == key:
    #si es un leaftnode
    if currentNode.leftnode == None and currentNode.rightnode == None:
      parentNode = currentNode.parent
      if parentNode.leftnode != None and parentNode.leftnode.key == key:
        parentNode.leftnode = None
      elif parentNode.rightnode != None and parentNode.rightnode.key == key:
        parentNode.rightnode = None

    #si es un nodo con un solo hijo
    elif currentNode.leftnode == None:
      currentNode.key = currentNode.rightnode.key
      currentNode.value = currentNode.rightnode.value
      currentNode.parent = currentNode.rightnode.parent
```

```
    elif currentNode.rightnode == None:
      currentNode.key = currentNode.leftnode.key
      currentNode.value = currentNode.leftnode.value
      currentNode.parent = currentNode.leftnode.parent

    #si tiene dos hijos
    else:
      tempNode = findMinNode(currentNode.rightnode)
      currentNode.key = tempNode.key
      currentNode.value = tempNode.value
      currentNode.parent = tempNode.parent
      deleteNode(currentNode.rightnode,currentNode.key)


  if currentNode.key < key:
    deleteNode(currentNode.rightnode,key)
  else:
    deleteNode(currentNode.leftnode,key)

"""DeleteKey"""

def deleteKey(B,key):

  if B == None:
    return None
  deleteNode(B.root,key)
  reBalance(B)
  return
```

Ejercicio 6

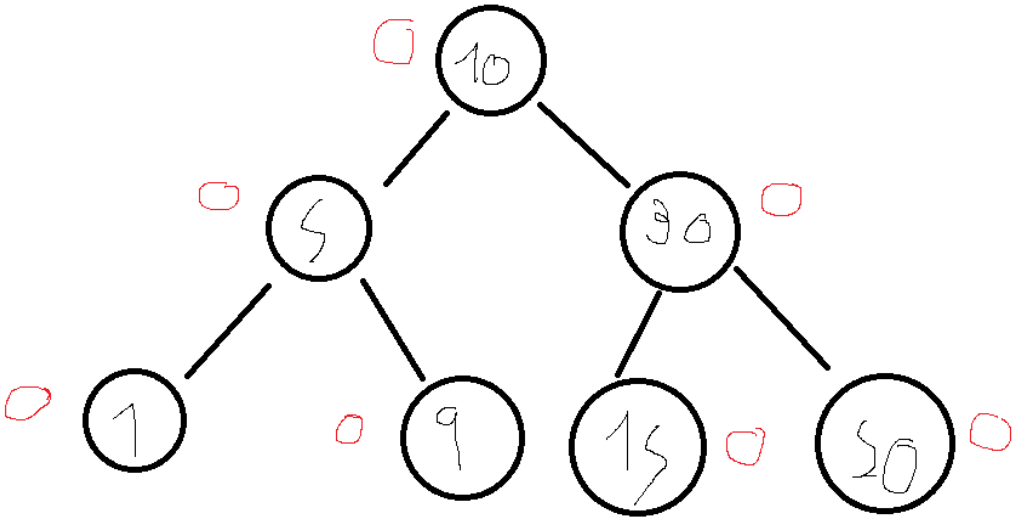
a) En un AVL el penultimo nivel tiene que estar completo.
Falso



Ejemplo donde un árbol no esta completo en el penúltimo nivel y aun asi cumple la condición de AVL

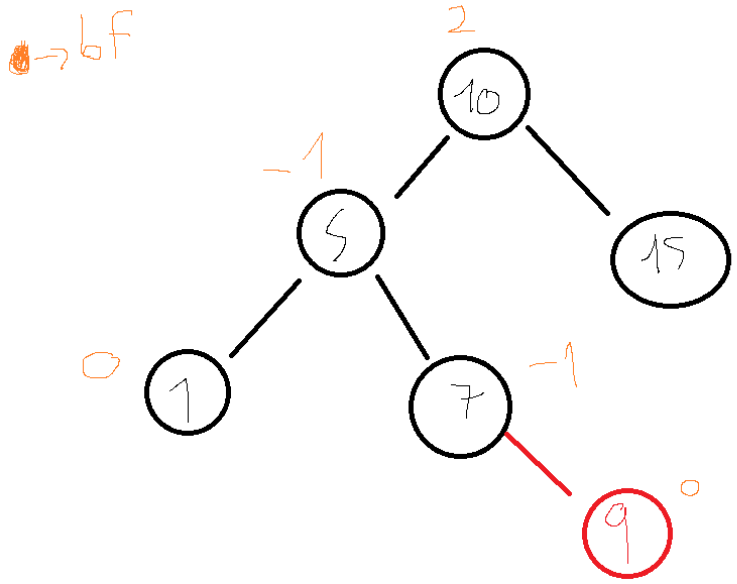
b)Un AVL donde todos los nodos tengan factor de balance 0 es completo

Verdadero



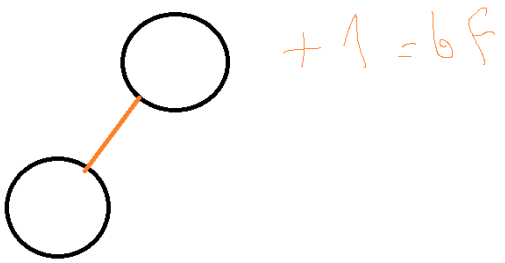
necesariamente el balance factor va a ser 0 ya que para que este sea cero necesita tener dos hijos o ninguno y el árbol completo cumple con esta condición

c)En la inserción en un AVL, si al actualizarle el factor de balance al padre del nodo insertado éste no se desbalanceó, entonces no hay que seguir verificando hacia arriba porque no hay cambios en los factores de balance.
Falso



en la imagen podemos apreciar que se agrego un nuevo nodo, el 9 y no afecto al balance factor de su padre pero podemos apreciar que la raiz quedo totalmente desbalanceada

d) En todo AVL existe al menos un nodo con factor de balance 0
Falso
Ya que los leaf node no cuentan puede haber un AVL con balance facto distinto de cero



Ejercicio 7
Existen 3 Casos
Caso1:
A y B tienen la misma altura o difieren en uno, en este caso x seria la raiz de nuestro nuevo árbol y tendría como hijo al árbol izquierdo A y el árbol derecho B
Caso2:
que el árbol de la izquierda (A) sea mas grande que el derecho(B) y que al armarlo quedaría desbalanceado, por ende buscamos en A un subárbol del mismo tamaño que B y luego aplicamos el algoritmo como en el caso 1
Caso3:
lo mismo que el caso 2 solamente que el árbol derecho es mayor que el izquierdo.
Ejercicio 8
Supongamos que existe una rama truncada en un AVL de altura h con longitud k, donde k < h/2. Entonces, la altura del subárbol izquierdo de ese nodo truncado es a lo sumo h/2-k-1, y la altura del subárbol derecho es a lo sumo h/2.

Por lo tanto, la diferencia de altura entre ambos subárboles es de al menos k+1. Pero esto contradice la propiedad de los AVL de que la diferencia de altura entre sus subárboles izquierdo y derecho es a lo sumo 1. Por lo tanto, no puede existir una rama truncada en un AVL de altura h con longitud menor que h/2,lo que demuestra que esa es la longitud mínima posible