**LAB8**

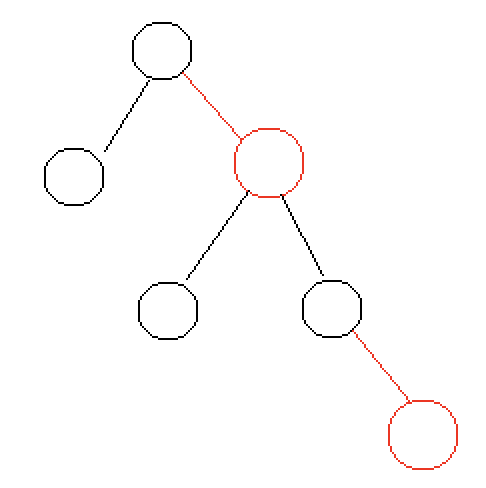
**Jimmy Palma**

**610756**

1. **An *AVL Tree* is a BST that satisfies a different balance condition, namely:**

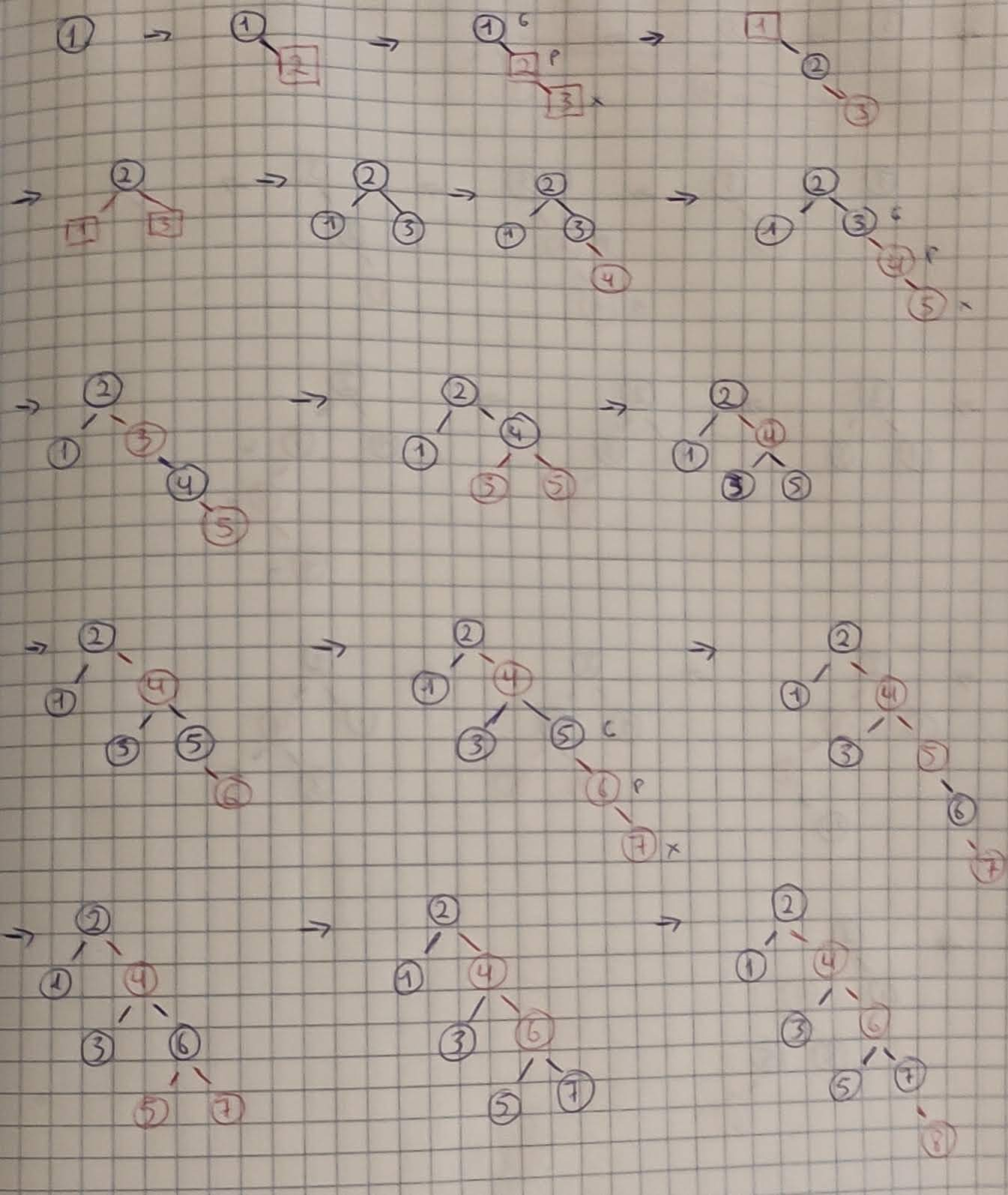
**The AVL Balance Condition For each internal node x, the height of the left child of x differs from the height of the right child of x by at most 1. (Equivalently, the heights of the left and right subtrees of x differ by at most 1.)**

**Create a red-black tree that does *not* satisfy the AVL Balance Condition.**

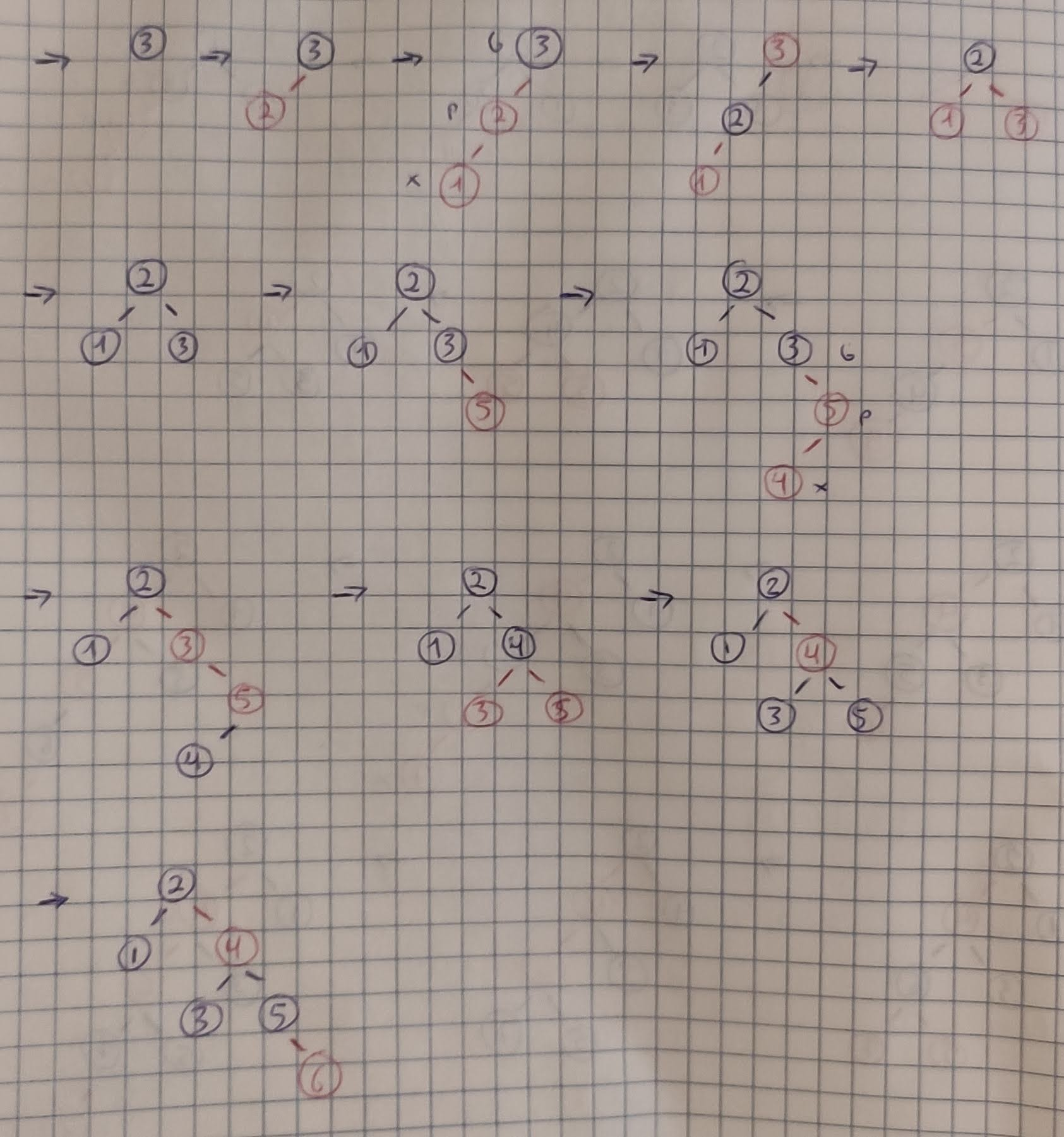


1. **Use the insertion algorithm for red-black trees to successively insert the following nodes, starting with an empty tree.**

**1, 2, 3, 4, 5, 6, 7, 8**



**3, 2, 1, 5, 4, 6**



**Note on Part (a): Recall that an already sorted insertion sequence is a worst case for an ordinary BST. Notice how the red-black balancing operations handle this to remain balanced.**

1. **Devise an algorithm IsPrime(*n*) which outputs TRUE if n is prime, FALSE otherwise. Then implement as a Java method. What is the asymptotic running time of IsPrime? Explain.**

Algorithm isPrime

Input: n num to validate

Output: true/ false

For i = 1 to n do

If i has more than 2 divisor then

Return false

Return true

///////// code //////////

private static boolean isPrime(int i) {  
 for (int j = 1; j <= i; j++) {  
 if (i % j == 0 && j != 1 && j != i) {  
 return false;  
 }  
 }  
 return true;

}