# Assignment 7 – AVL-trees and B-trees

Myo Thet Tun - 2918551

Copy the relevant code piece and also show all your work on this document and upload only this document.

**Question 1**

Using the class BinarySearchTree from last week add the following methods:

rotateLeftRoot() - rotate tree left at root preserving the tree ordering.

rotateRightRoot() - rotate tree right at root preserving the tree ordering.

rotateRightAt(E x) - rotate tree right at node x preserving the tree ordering.

E predecessor(E x) – find the predecessor of x in the tree.

**public** **void** rotateLeftRoot() {

BNode<E> k1 = root;

BNode<E> y = root.right().left();

BNode<E> k2 = root.right();

k1.setRight(y);

k2.setLeft(k1);

root = k2;

}

**public** **void** rotateRightRoot() {

BNode<E> k2 = root;

BNode<E> y = root.left().right();

BNode<E> k1 = root.left();

k1.setRight(k2);

k2.setLeft(y);

root = k1;

}

**public** E predecessor(E x) {

BNode<E> parent = findPredecessor(root, x);

**if** (parent == **null**){

System.***out***.println("null");

**return** **null**;

}

**return** parent.data();

}

**public** BNode<E> findPredecessor(BNode<E> parent, E x) {

// BNode<E> temp = null;

**if** (parent.left().data() == x || parent.right().data() == x ) {

**return** parent;

}

**else**{

findPredecessor(parent.left(), x);

findPredecessor(parent.right(), x);

**return** parent;

}

}

**public** **void** rotateRightAt(E x) {

BNode<E> pd = findPredecessor(root, x);

BNode<E> k2 = pd;

BNode<E> y = pd.left().right();

BNode<E> k1 = pd.left();

k1.setRight(k2);

k2.setLeft(y);

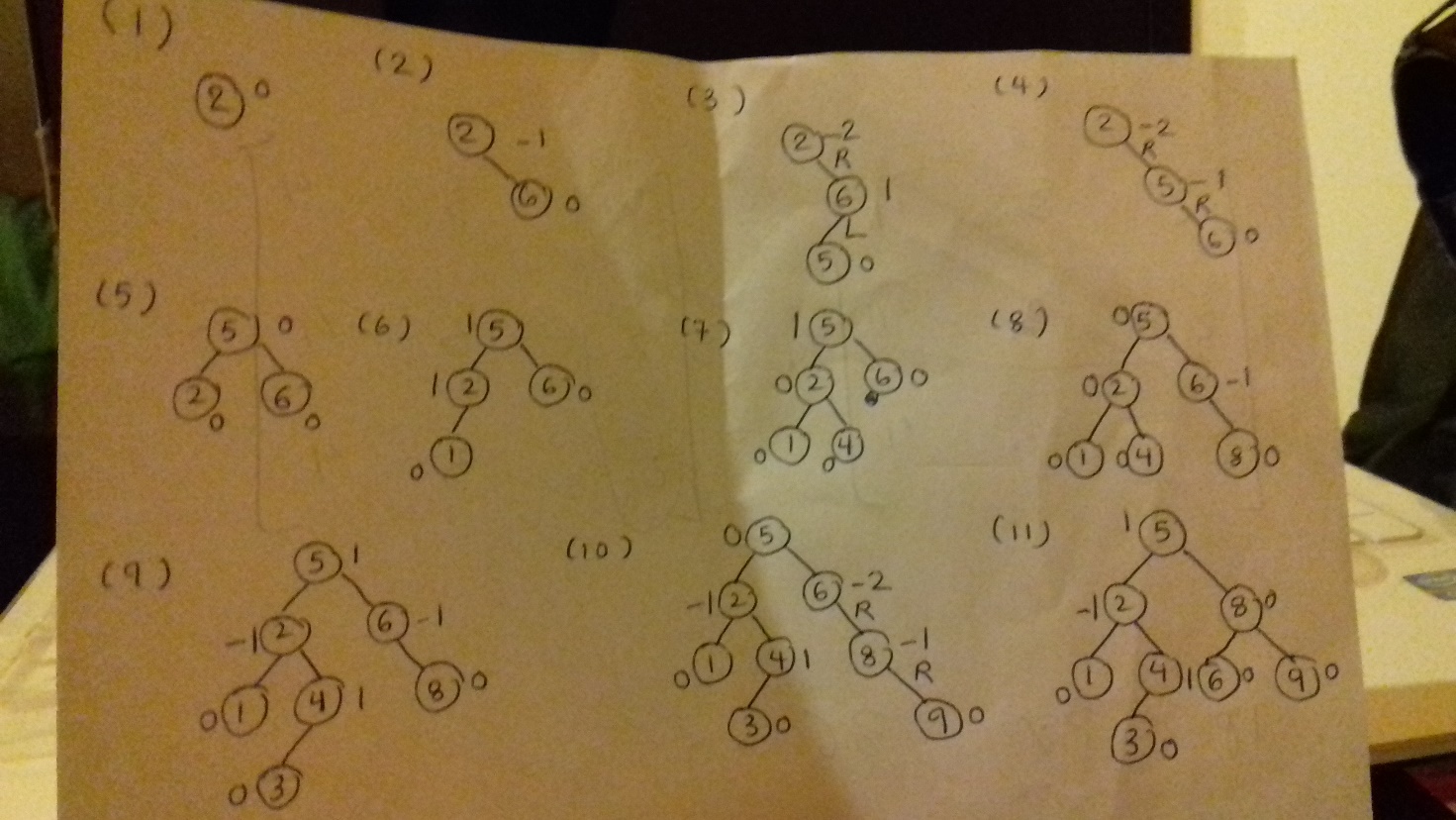
root = k1;

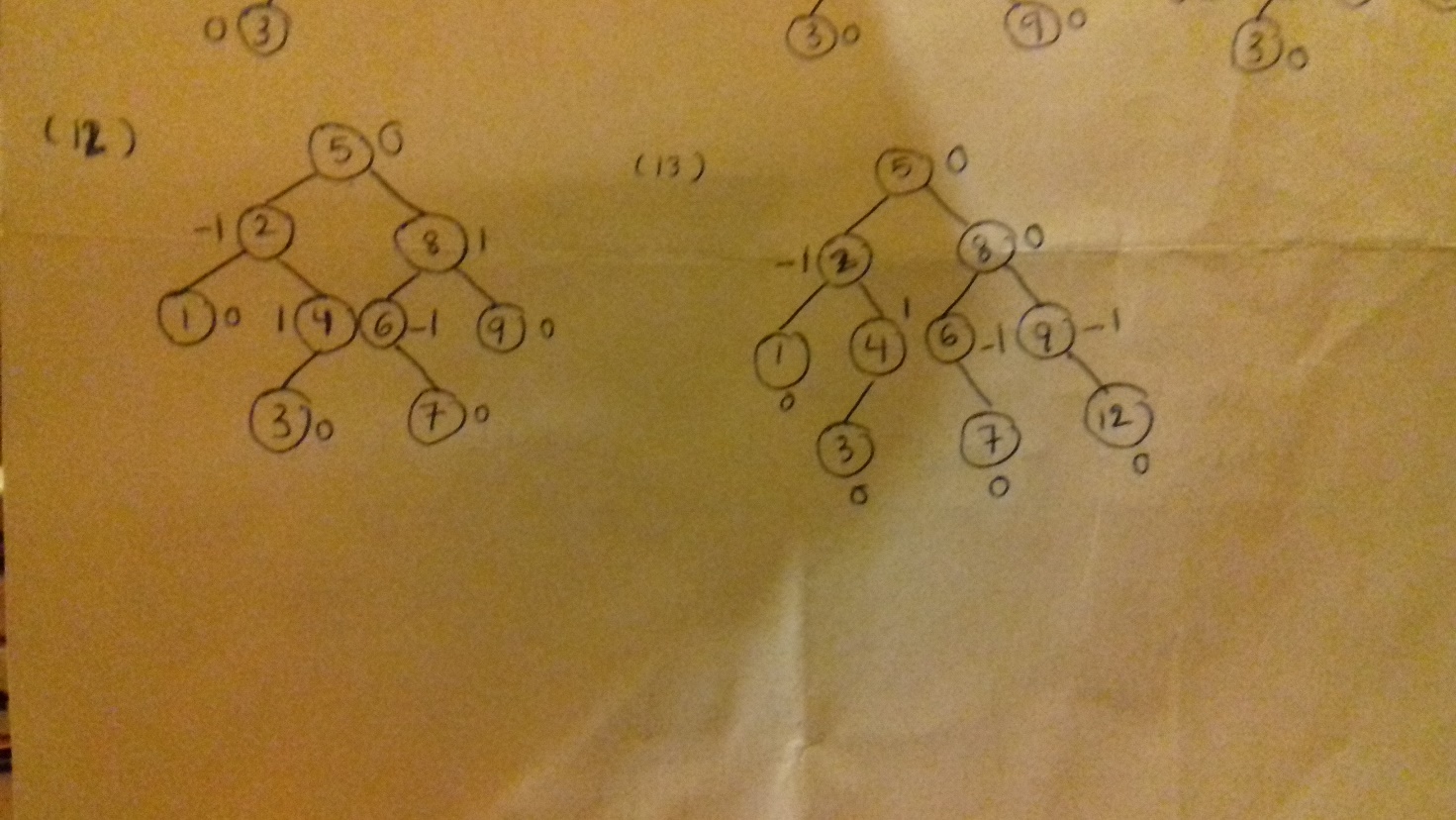
}

For the remaining questions, it is important to show all the steps. For the AVL-tree questions, you must show the rotations involved as the tree grows. Remember you must preserve the invariant that the tree is a binary search tree where the height of the left sub-tree differs from that of the right sub-tree by no more than 1.

**Question 2**

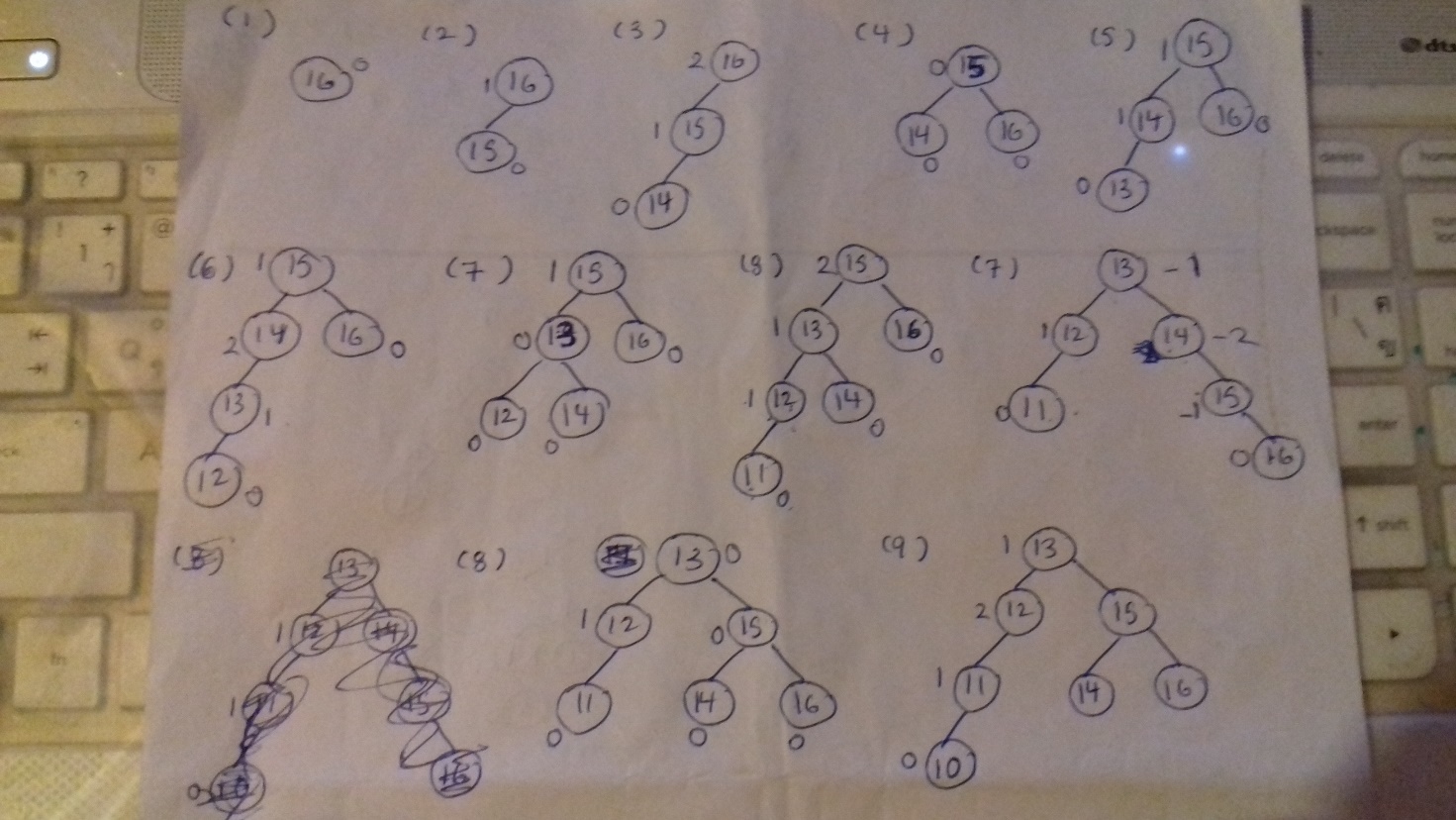
Construct an AVL-tree from the given list of values: 2, 6, 5, 1, 4, 8, 3, 9, 7, 12. Calculate and show the balance factors of each node after each insert operation.

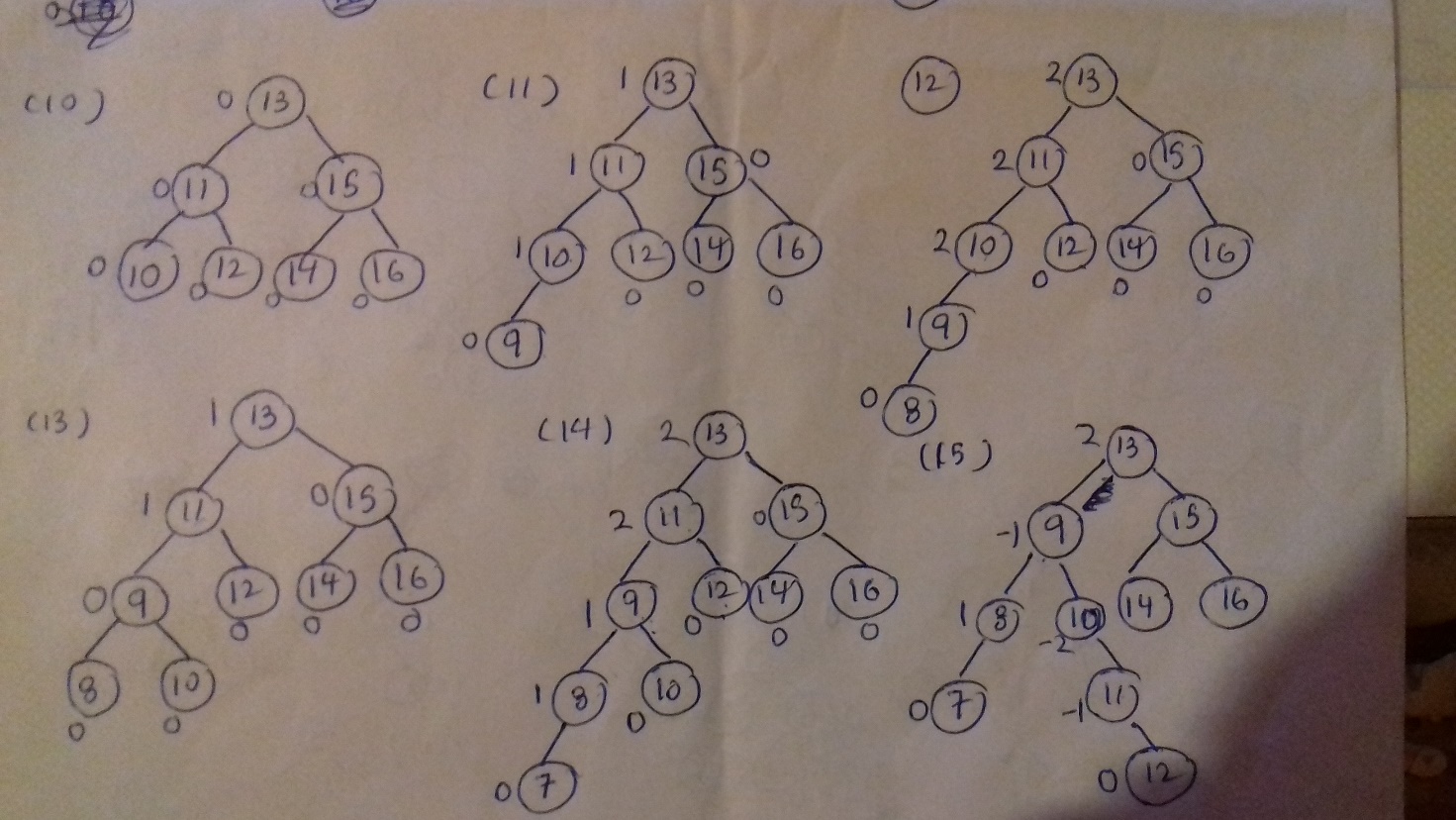


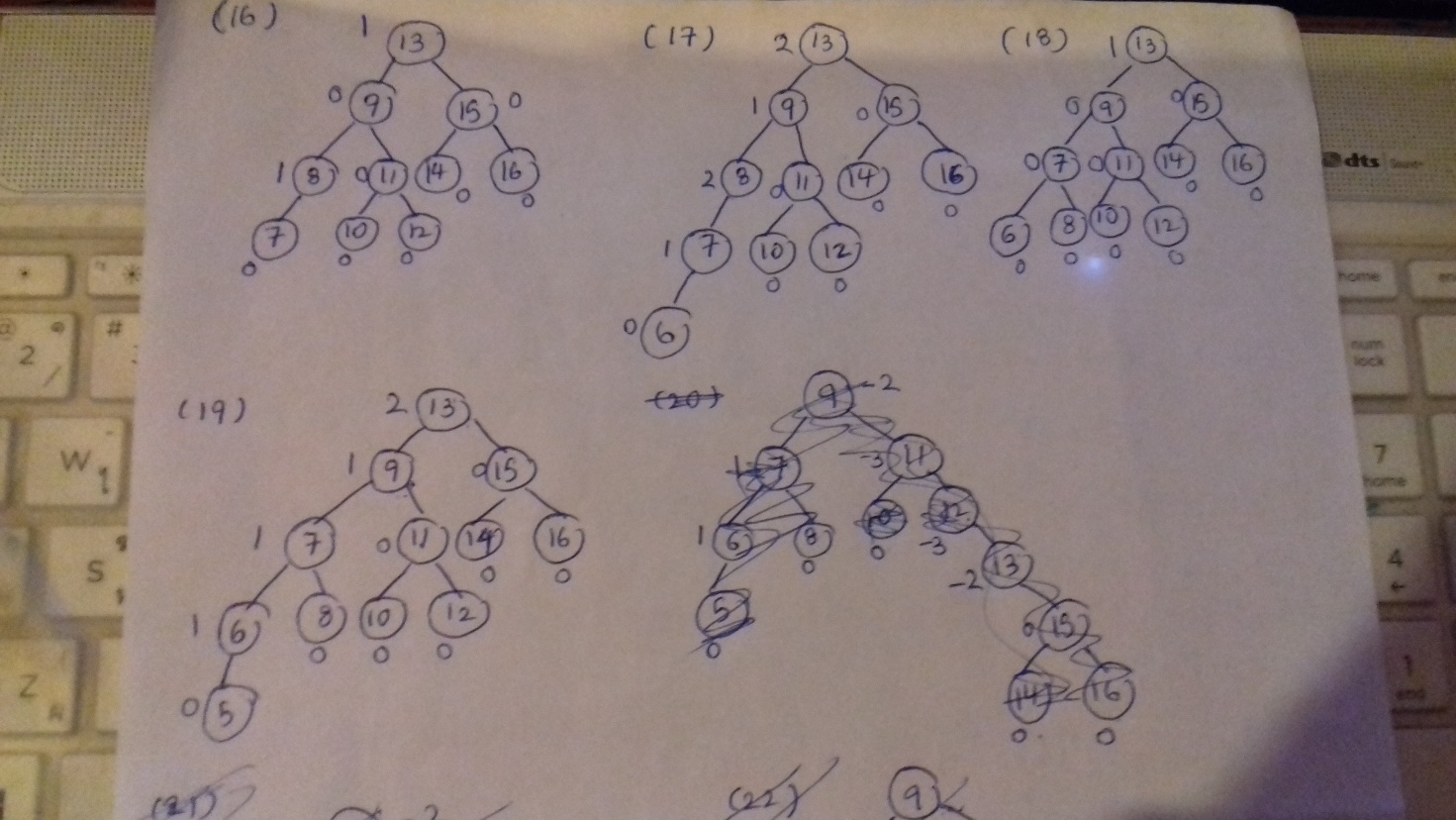


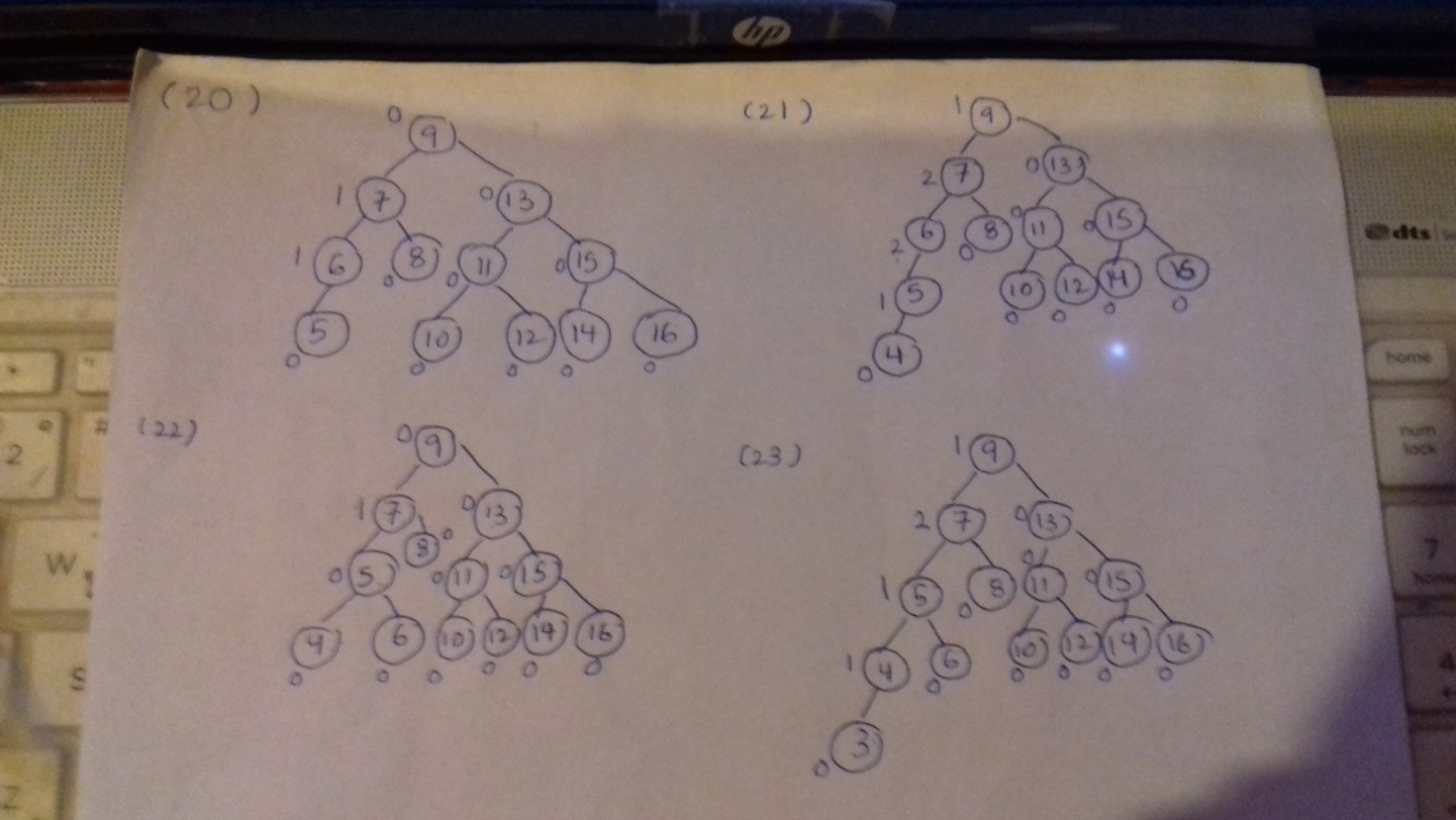
**Question 3**

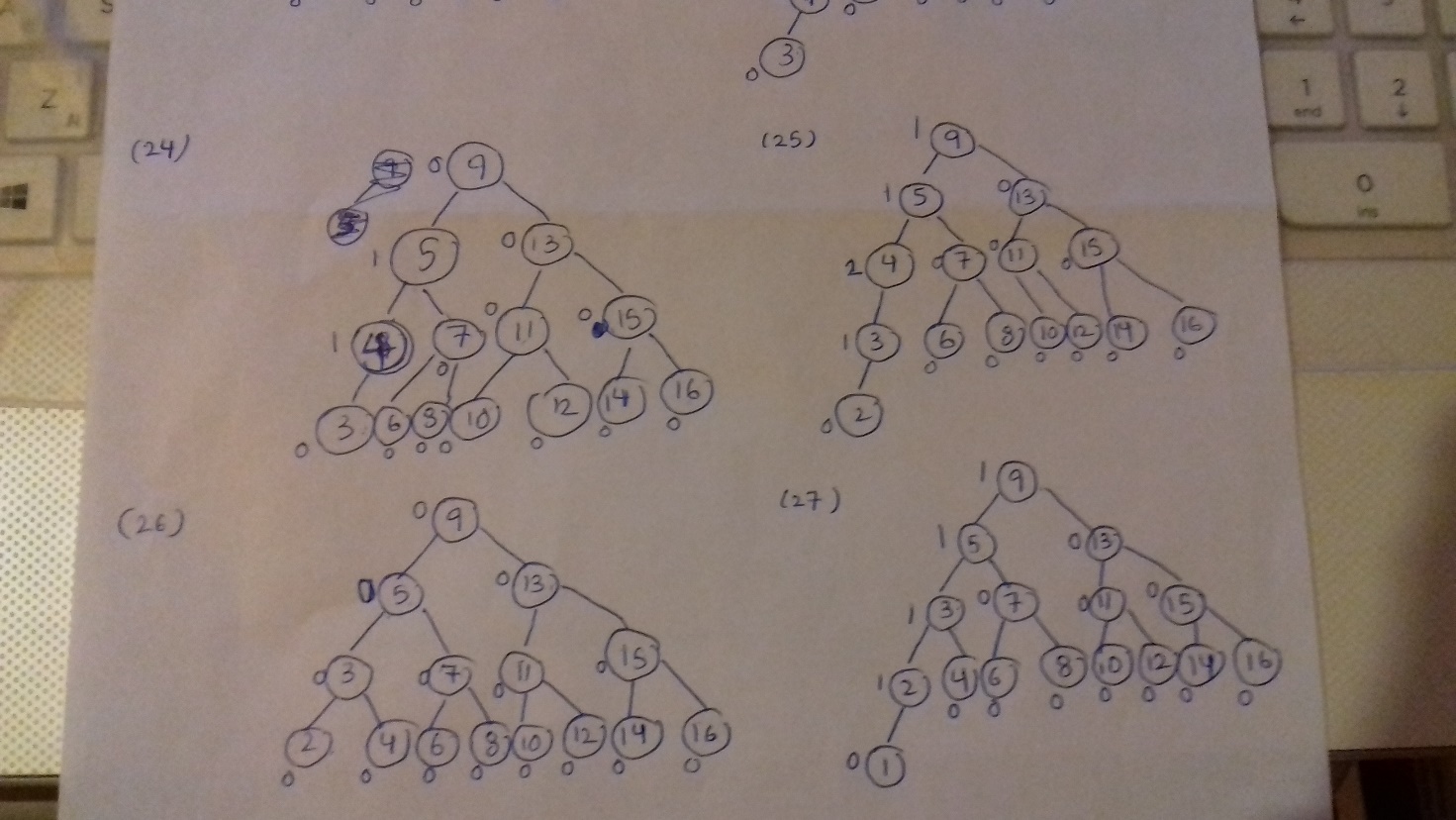
Construct an AVL-tree from the following list of values: 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.











**Question 4**

Construct a B-tree with a maximum of 4 items per node for the list of numbers: 2, 11, 3, 7, 8, 10, 9, 17, 21, 1, 12, 18. You must show the construction of the tree as part of your answer.

