HS-Bremerhaven
Prof. Dr. Peter Kelb

Algorithmen und Datenstrukturen
(1. Versuch)

WS 2012/2013
30.1.2013

Name:	Maximale Punktzahl: 40
Vorname:	Erreichte Punktzahl:
Matrikel:	Note:

1. Gegeben ist die folgende (Teil-)Implementierung eines Patricia Trees: 15

```
class PatriciaTree {

    class Node {

        public Node(char key,int bitPos,Node succ) ...
        public Node(char key,int bitPos) ...

        public char m_Key;
        public int m_BitPos;
        public Node m_Left;
        public Node m_Right;
    }

    private Node m_Root = null;
```

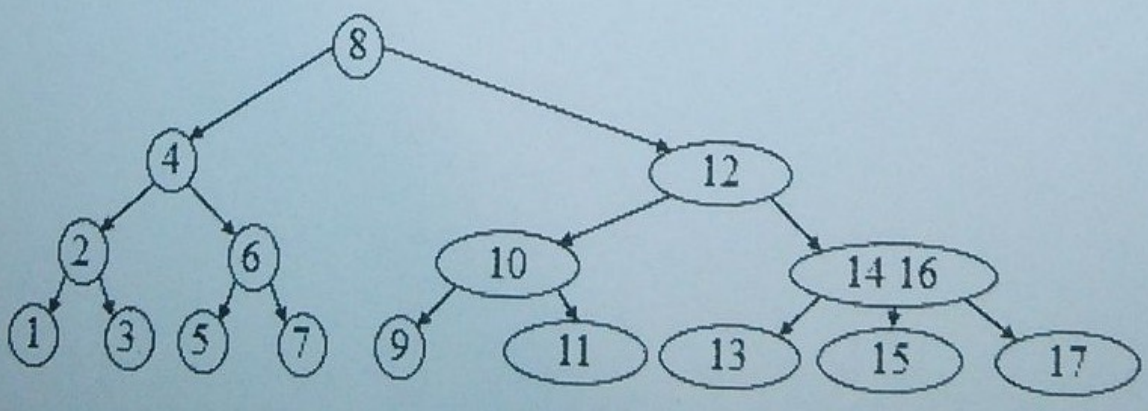
Implementieren Sie eine Objektmethode `convert()`, die einen Vector zurückliefert, der alle Elemente des Baums enthält. Verwenden Sie die add Methode der Vector Klasse, um ein Element dem Vektor hinzuzufügen.

2. Gegeben ist die folgende (Teil-)Implementierung eines Rot-Schwarz-Baums: 15

```
public class BlackRedTree<K extends Comparable<K>,D> {

    class Node {
        public Node(K key,D data)  ...

        K m_Key;
        D m_Data;
        Node m_Left = null;
        Node m_Right = null;
        boolean m_bIsRed = true;
    }
    private Node m_Root = null;
```

Implementieren Sie eine Objektmethode `depth()`, die die Tiefe des Baums zurückliefert. Dabei werden die roten Kanten nicht gezählt sondern nur die schwarzen Kanten.

3. Gegeben ist der folgende Top-Down-2-3-4 Baum. Zeichnen Sie den entstehenden Baum nach dem Löschen der 8. 10



1)

```
Vector convert(){
    ~~Vector v~~
    Vector<Node> v = new Vector();
    a(m_Root, -1, v);
    return v;
}

void a(Node n, int dadPos, Vector<Node> v){
    if(n != null && n.m_BitPos > dadPos){
        v.add(n);
        a(n.m_Left, n.m_BitPos, v);
        a(n.m_Right, n.m_BitPos, v);
    }
}
```

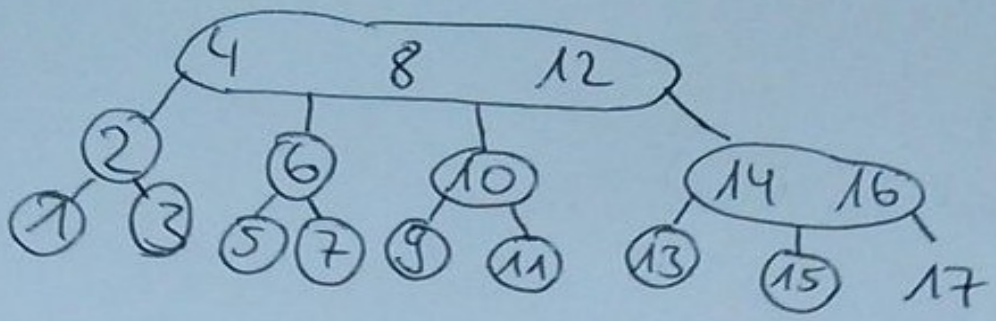
2)

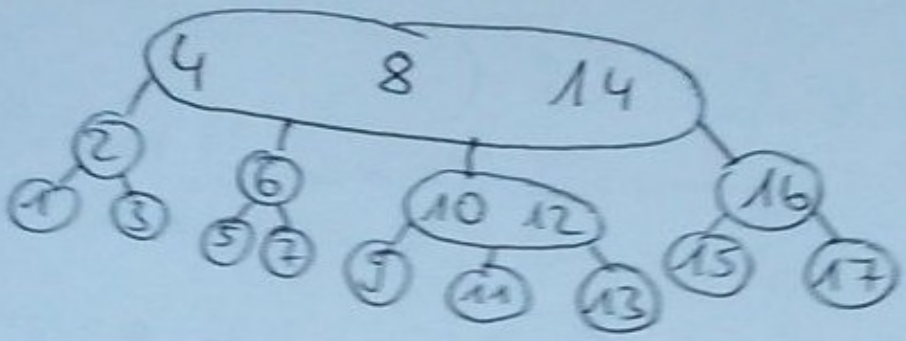
```
int depth(){
    return depth(m_Root);
}

int depth(Node n){
    if(n != null){
        if(!n.m_blsRed && (
            (n.m_Left == null || !n.m_Left.m_blsRed &&
            n.m_Right == null || !n.m_Right.m_blsRed))){
            return 1 + Math.max(depth(n.m_Left), depth(n.m_Right));
        }
        return Math.max(depth(n.m_Left), depth(n.m_Right));
    }
    return 0;
}
```

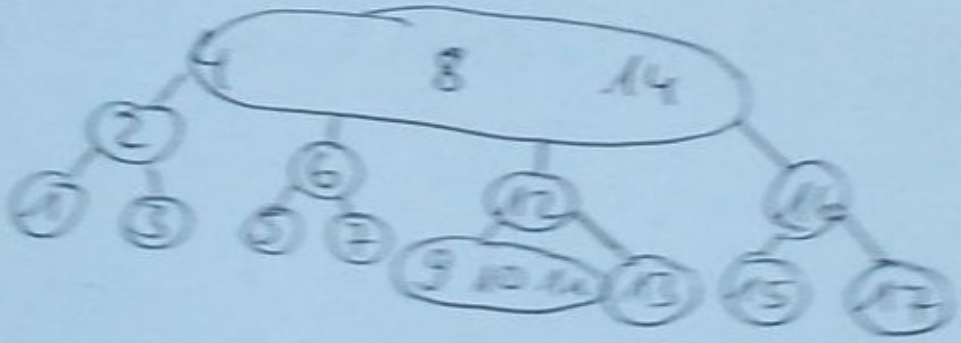
Komma

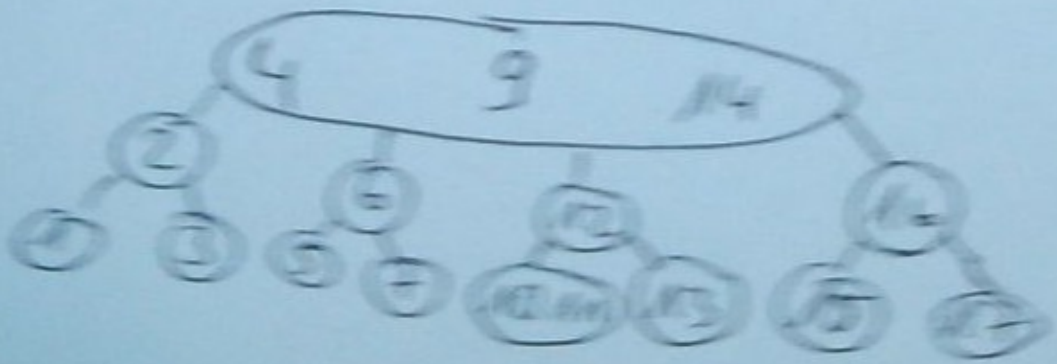
Komma

3)


4 8 12
2 6 10 14 16
1 3 5 7 9 11 13 15 17


4 8 14
2 6 10 12 16
1 3 5 7 9 11 13 15 17


8 14
2 6 12 16
1 3 5 7 9 10 11 13 15 17


4 9 14
2 6 12 16
1 3 5 7 10 11 13 15 17