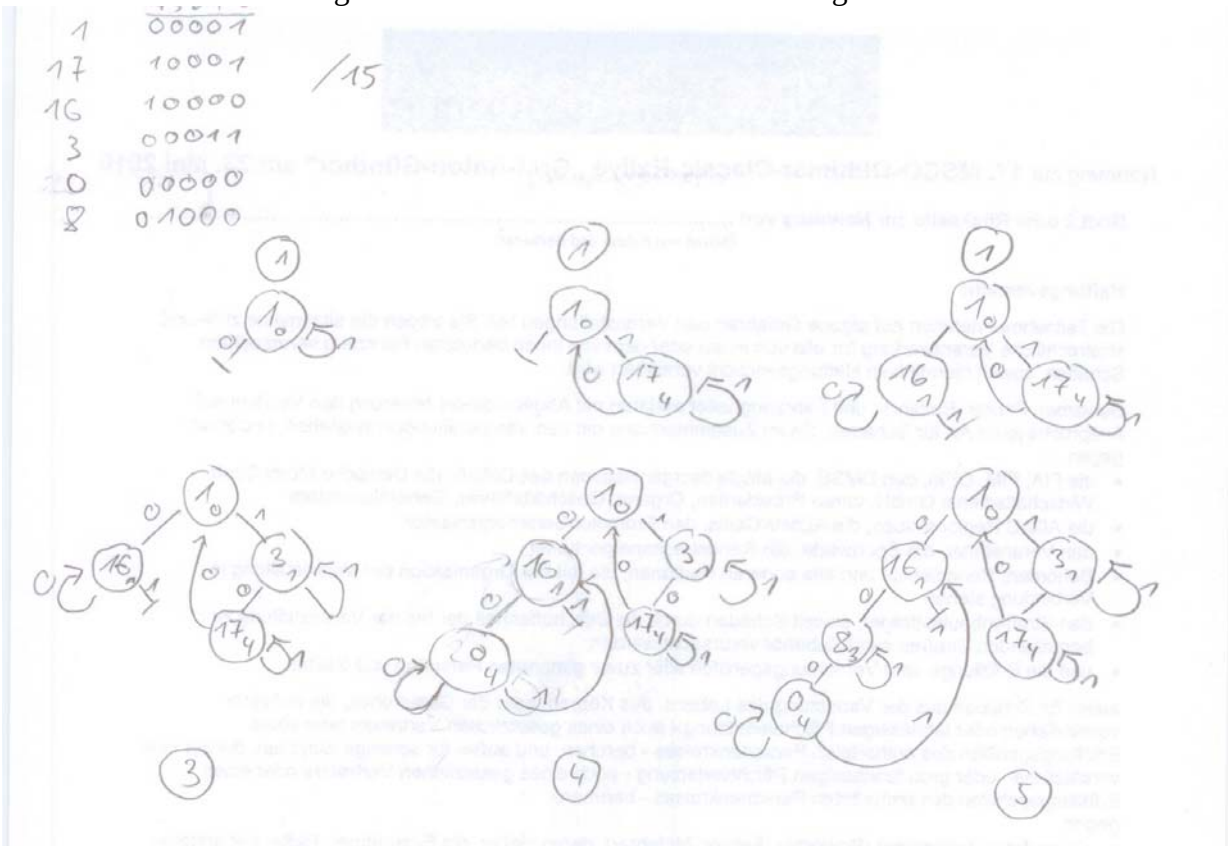
Name: Pseudonym: Studiengang: (bitte unterstreichen)
Vorname: Unterschrift: INF WINF SI
Matrikelnummer:

1. Konstruieren Sie zu der Zahlenfolge [1,17,16,3,0,8] den entstehenden Patrica Tree mit aufsteigenden Bitpositionen (an der Wurzel wird Bit 0 getestet). Zeichnen Sie alle Zwischenschritte auf. Vergessen Sie nicht die Kantenbeschriftung. / 15



2. Zeichnen Sie die Entwicklung der Hashtabelle auf, wenn Sie die Zahlenfolge [13,15,16,3,2,7,5,4,9] als Schlüssel einfügen. Die Anfangsgröße der Tabelle ist 5. Ein Rehashing erfolgt bei ≥80%. Die neue Größe wird aus der alten Größe `n` durch `f(n)=(n+1)*2-1`. Die Hashfunkion ist die Identität. Zeichnen Sie die Hashtabelle nach jedem Einfügen. / 15

   [_,_,_,13,_]
   [15,_,_,13,_]
   [15,16,_,13,_]
   [15,16,_,13,3]
   [_,_,13,3,15,16,_,_,_,_,_]
   [_,_,13,3,15,16,2,7,_,_,_]
   [_,_,13,3,15,16,2,7,5,_,_]
   [_,_,13,3,15,16,2,7,5,4,_]
   [_,_,2,3,4,5,_,7,_,9,_,_,_,13,_,15,16,_,_,_,_,_,_]

3. Gegeben sei der folgende Ausschnitt der Klasse für Rot-Schwarz Bäume. Implementieren Sie eine Methode `cnt3Nodes`, die die Anzahl der 3er Knoten im Baum ermittelt. / 15

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

   class Node {
…
         K m_Key;
```

```
        D m_Data;
        NodeRef m_Left = new NodeRef();
        NodeRef m_Right = new NodeRef();
        boolean m_bIsRed = true;
    }

    class NodeRef {
        public Node get() {     return m_Node;     }
        private Node m_Node = null;
    }

    public int cnt3Nodes() {                                    3
        return cnt3Nodes(m_Root.get());
    }

    private int cnt3Nodes(Node n) {
        if (n == null)
            return 0;
        boolean is3Node = (n.m_Left.get() != null &&            8
                           n.m_Left.get().m_bIsRed)
                        ^ (n.m_Right.get() != null &&
                           n.m_Right.get().m_bIsRed);
        return cnt3Nodes(n.m_Left.get())                        4
             + cnt3Nodes(n.m_Right.get())
             + (is3Node ? 1 : 0);
    }
…
    private NodeRef m_Root = new NodeRef();
}
```

4. Das Distribution Counting ist das schnellste Sortierverfahren. Warum wird es nicht oft eingesetzt? / 5

   Das Distribution Counting kann nur eingesetzt werden, wenn Zahlen aus einem kleinen Wertebereich sortiert werden. Dies ist selten gegeben.