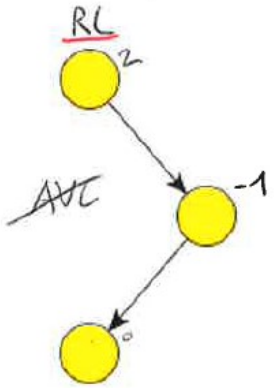
AVL Bäume Trainingsaufgaben – Lösungen

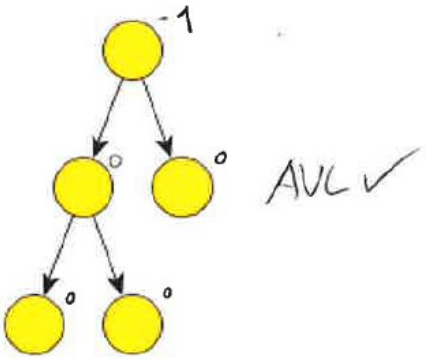
1. AVL-Baum Validierung

a.)

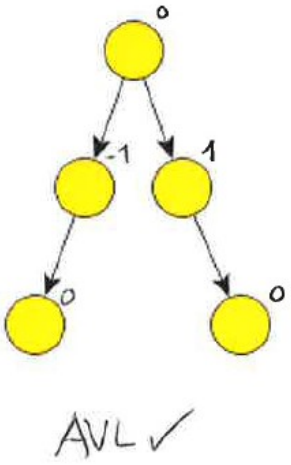
b.)



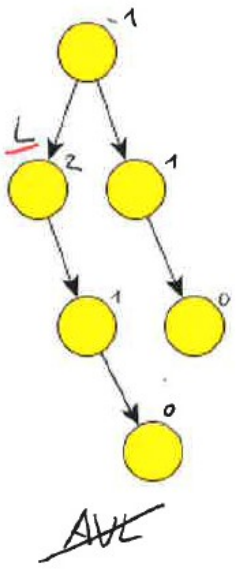
c.)



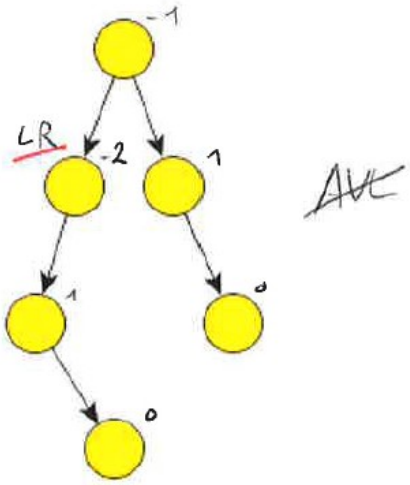
d.)



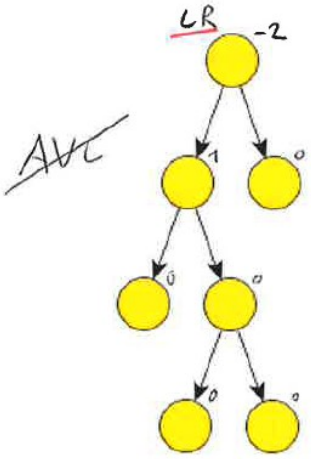
e.)



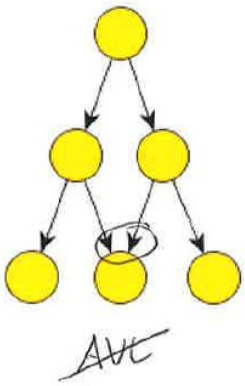
f.)



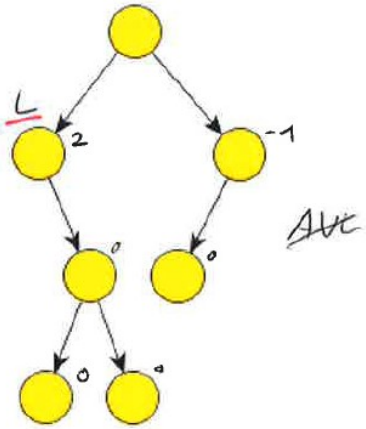
g.)



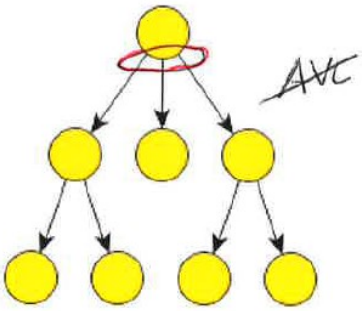
h.)



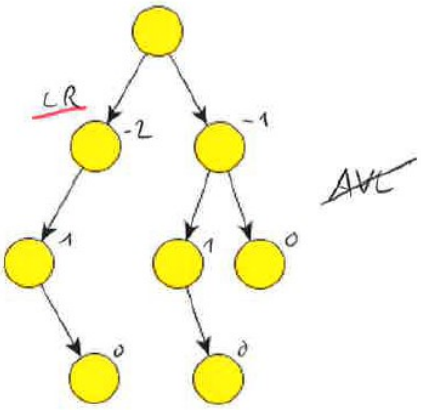
i.)



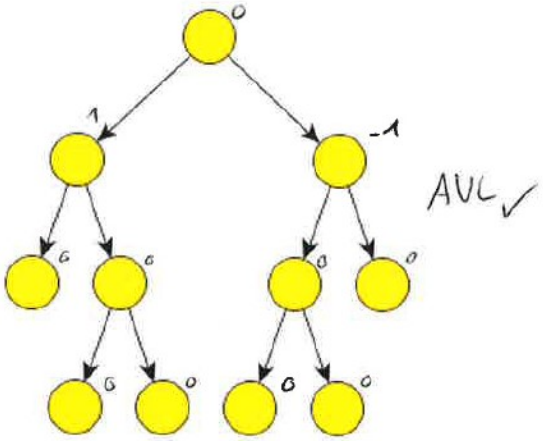
j.)



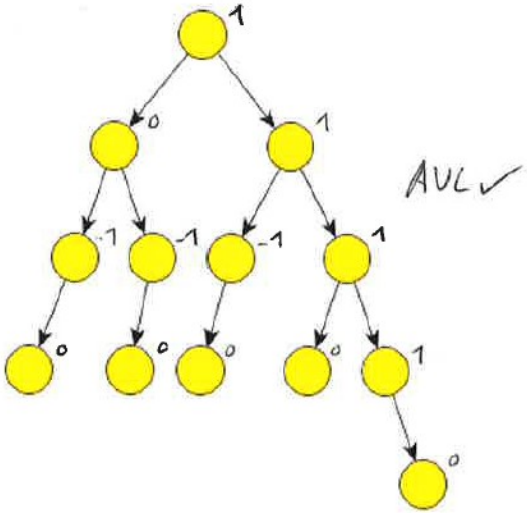
k.)



l.)

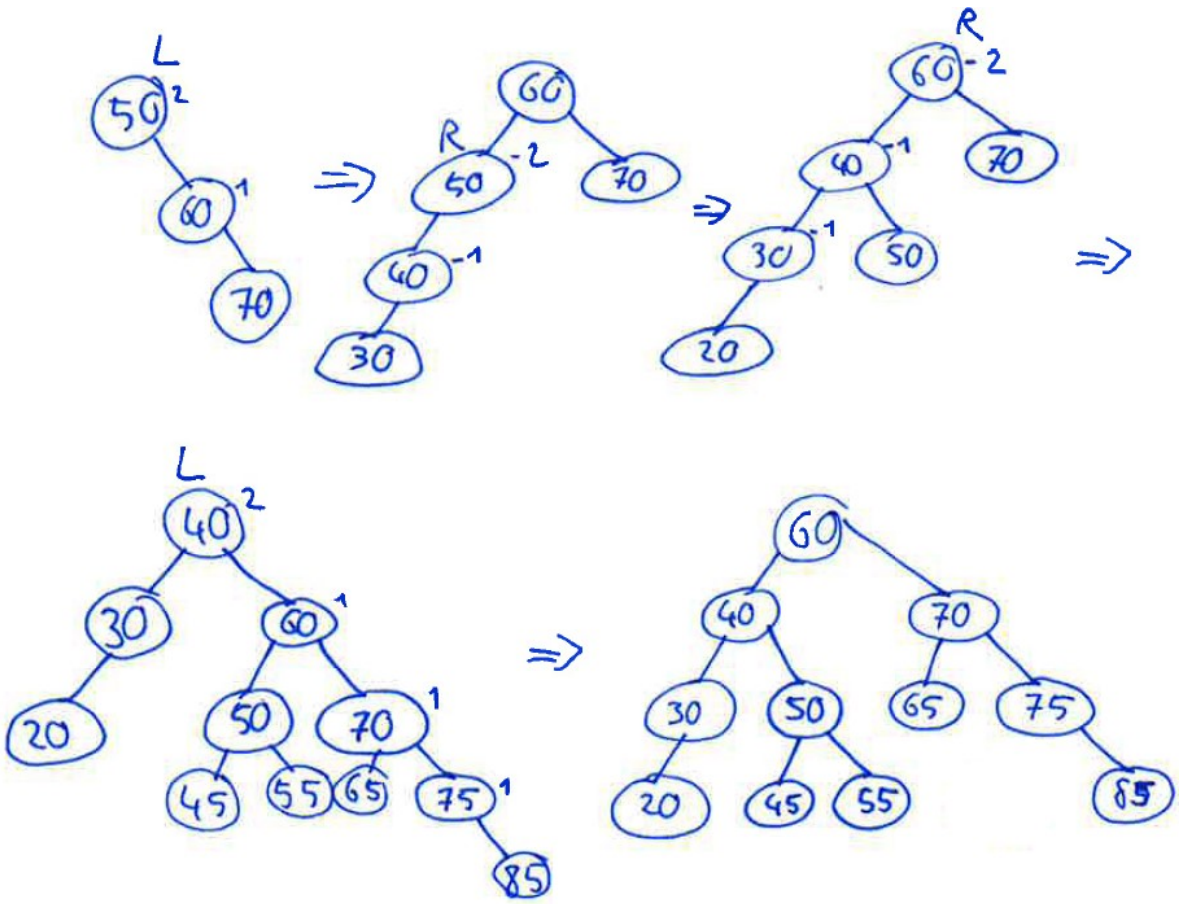


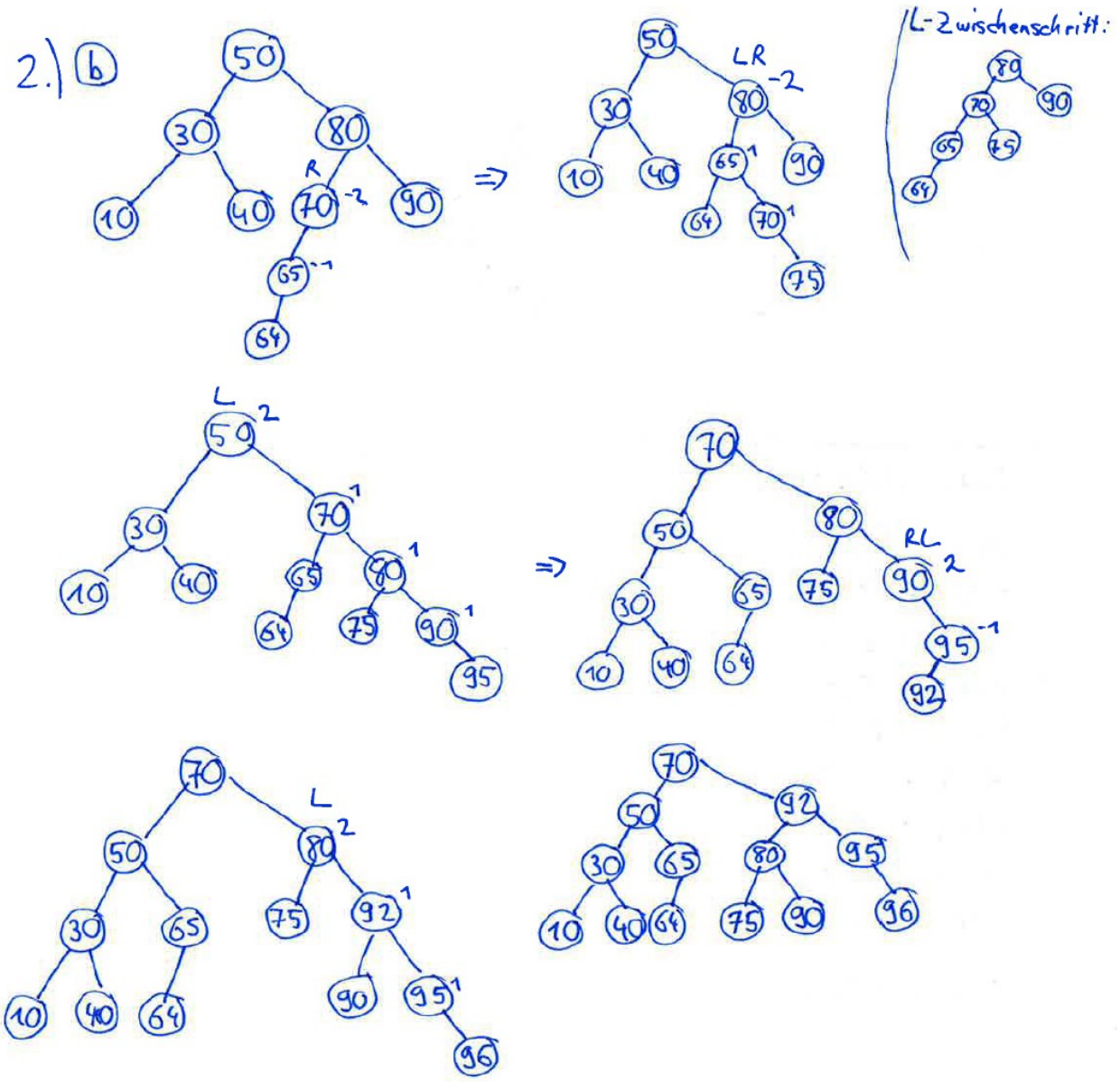
m.)



2. Hinzufügen von Knoten in einen AVL-Baum

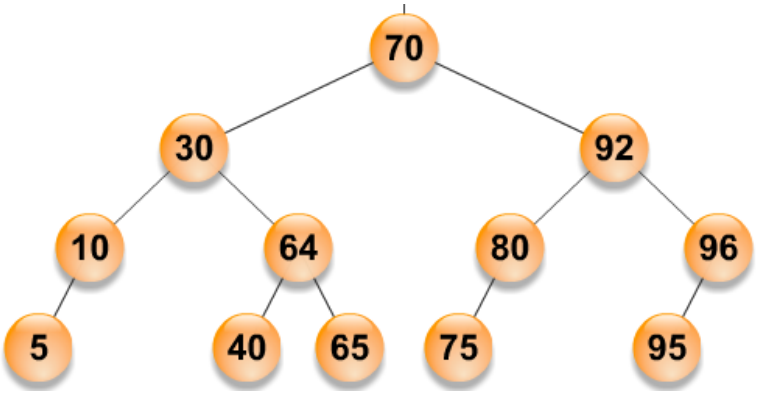
a.)



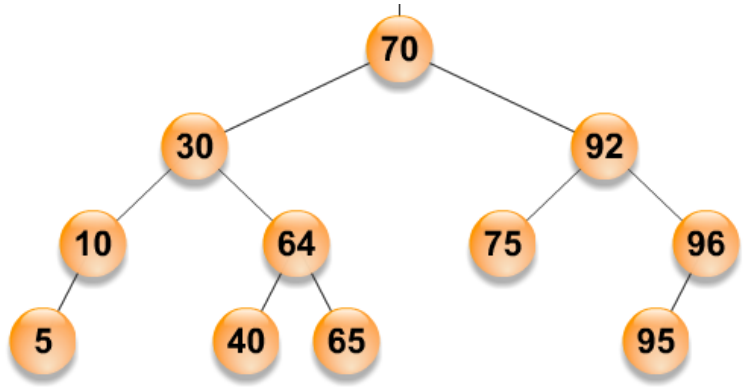


3. Löschen von Knoten aus einem AVL-Baum

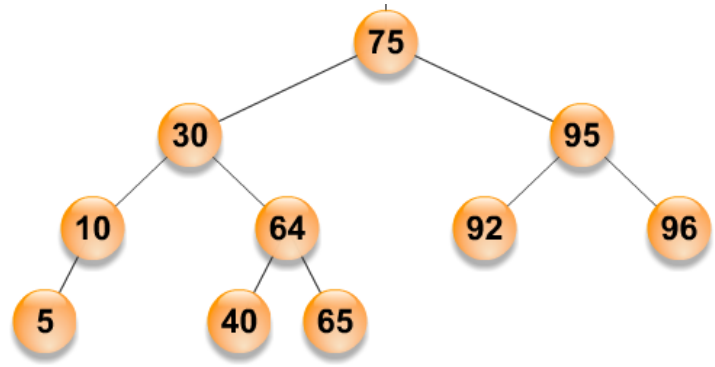
a.) Nach dem Entfernen von: 50



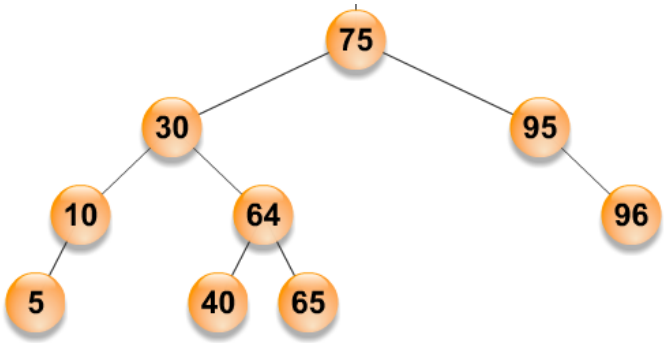
Nach dem Entfernen von: 80



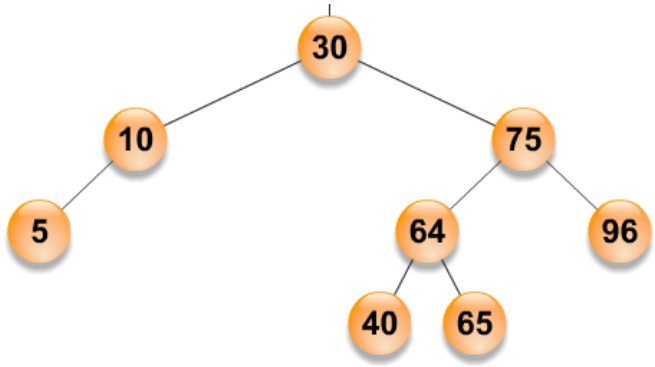
Nach dem Entfernen von: 70



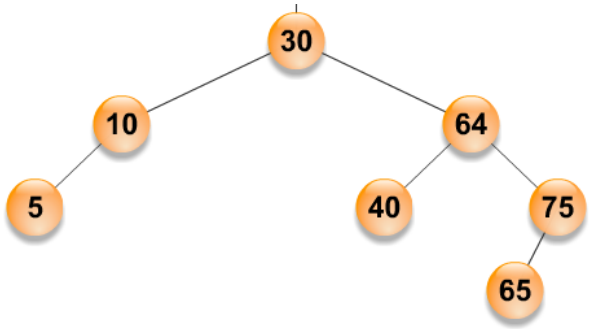
Nach dem Entfernen von: 92



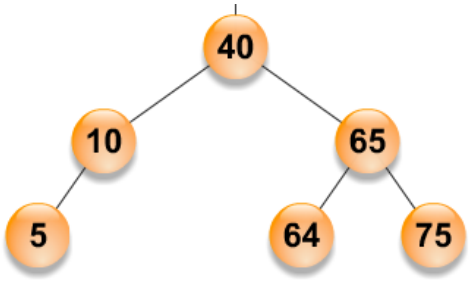
Nach dem Entfernen von: 95



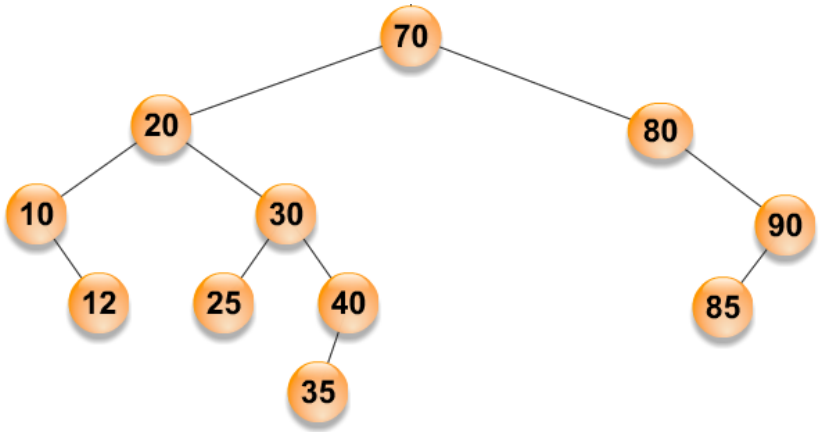
Nach dem Entfernen von: 96



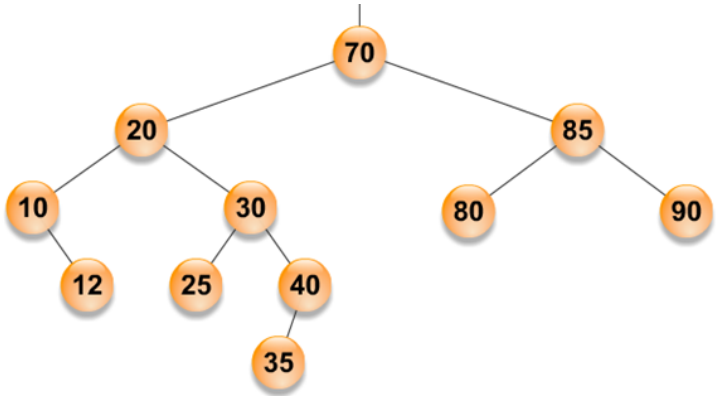
Nach dem Entfernen von: 30



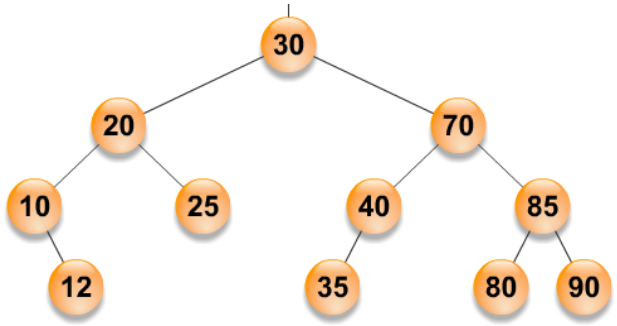
b.) Wurzel-Element entfernen:



Ausbalancieren (RL-Rotation bei 80)



Zweites Ausbalancieren (LR-Rotation bei 70)



4. Programmieraufgaben

a.) Elemente in absteigender Ordnung ausgeben

```
public void printDescendingOrder() {
  StringBuilder sb = new StringBuilder();
  sb.append("Descending Order: ");
  printDescendingOrder(root, sb);
  System.out.println(sb.toString());
}

private void printDescendingOrder(Node<K, E> node, StringBuilder sb) {
  if (node == null) {
    return;
  }
  printDescendingOrder(node.right, sb);
  sb.append(node.key + " | ");
  printDescendingOrder(node.left, sb);

}
```

b.) Validierungsmethode

```
void validateOrder() {
  checkRange(root, null, null);
}

private void checkRange(Node<K, E>  n, K min, K max) {
  if (n != null) {
    if ((min != null && min.compareTo(n.key) >= 0)
      || (max != null && max.compareTo(n.key) <= 0))
          throw new IllegalStateException();

    checkRange(n.left, min, n.key);
    checkRange(n.right, n.key, max);
  }
}
```