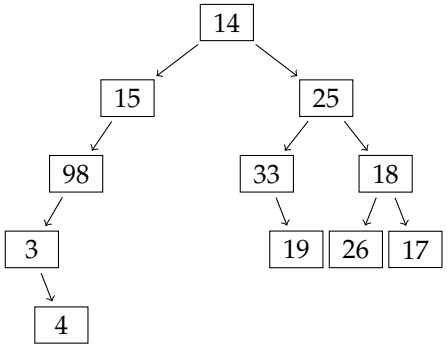
# Aufgabe 3

(a) Betrachten Sie folgenden Binärbaum T.

Geben Sie die Schlüssel der Knoten in der Reihenfolge an, wie sie von einem Preorder-Durchlauf (= TreeWalk) von T ausgegeben werden.




**Exkurs: Preorder-Traversierung eines Baum**

besuche die Wurzel, dann den linken Unterbaum, dann den rechten Unterbaum; auch: WLR

```
62  private void besuchePreorder(BaumKnoten knoten,
    ↪  ArrayList<Comparable> schlüssel) {
63    if (knoten != null) {
64      schlüssel.add((Comparable) knoten.gibSchlüssel());
65      besuchePreorder(knoten.gibLinks(), schlüssel);
66      besuchePreorder(knoten.gibRechts(), schlüssel);
67    }
68  }
```

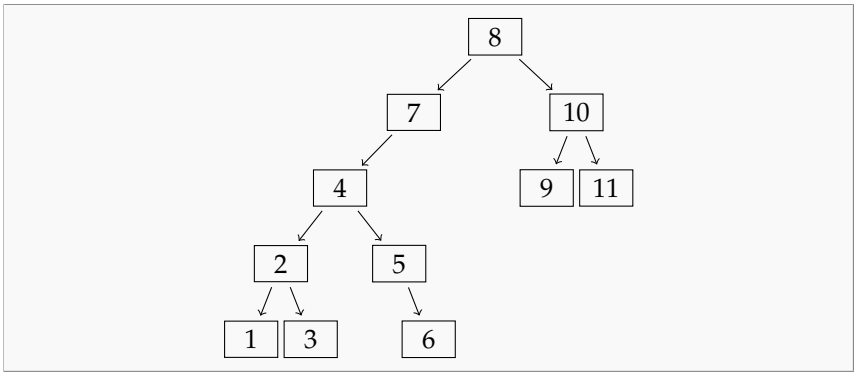
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/baum/BinaerBaum.java`

14, 15, 98, 3, 4, 25, 33, 19, 18, 26, 17

(b) Betrachten Sie folgende Sequenz als Ergebnis eines Preorder-Durchlaufs eines binären Suchbaumes $T$. Zeichnen Sie $T$ und erklären Sie, wie Sie zu Ihrer Schlussfolgerung gelangen.

$$[8,7,4,2,1,3,5,6,10,9,11]$$

**Hinweis:** Welcher Schlüssel ist die Wurzel von $T$? Welche Knoten sind in seinem linken/rechten Teilbaum gespeichert? Welche Schlüssel sind die Wurzeln der jeweiligen Teilbäume?




(c) Anstelle von sortierten Zahlen soll ein Baum nun verwendet werden, um relative Positionsangaben zu speichern. Jeder Baumknoten enthält eine Beschriftung und einen Wert (vgl. Abb. 1), der die ganzzahlige relative Verschiebung in horizontaler Richtung gegenüber seinem Elternknoten angibt. Die zu berechnenden Koordinaten für einen Knoten ergeben sich aus seiner Tiefe im Baum als $y$-Wert und aus der Summe aller Verschiebungen auf dem Pfad zur Wurzel als $x$-Wert. Das Ergebnis der Berechnung ist in Abb. 2 visualisiert. Geben Sie einen Algorithmus mit linearer Laufzeit in Pseudo-Code oder einer objektorientierten Programmiersprache Ihrer Wahl an. Der Algorithmus erhält den Zeiger auf die Wurzel eines Baumes als Eingabe und soll Tupel mit den berechneten Koordination aller Knoten des Baums in der Form (Beschriftung, $x$, $y$) zurück- oder ausgeben.

```java
public class Knoten {
  public Knoten links;
  public Knoten rechts;
  public String name;

  /**
   * Bewegung bezüglich des Vorknotens. Relative Lage.
   */
  public int xVerschiebung;

  public Knoten(String name, int xVerschiebung) {
    this.name = name;
    this.xVerschiebung = xVerschiebung;
  }

  public void durchlaufen() {
    durchlaufe(this, 0 + xVerschiebung, 0);
  }

  private void durchlaufe(Knoten knoten, int x, int y) {
    System.out.println("Beschriftung: " + knoten.name + " x: " + x +
    ↪  " y: " + y);

    if (links != null) {
      links.durchlaufe(links, x + links.xVerschiebung, y + 1);
    }
    if (rechts != null) {
      rechts.durchlaufe(rechts, x + rechts.xVerschiebung, y + 1);
    }
```

```
31      }
32
33      public static void main(String[] args) {
34        Knoten a = new Knoten("a", 1);
35        Knoten b = new Knoten("b", 1);
36        Knoten c = new Knoten("c", -2);
37        Knoten d = new Knoten("d", 2);
38        Knoten e = new Knoten("e", 0);
39
40        a.links = b;
41        a.rechts = c;
42        c.links = d;
43        c.rechts = e;
44
45        a.durchlaufen();
46      }
47    }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/Knoten.java`