

## 5. Programmieraufgabe Computerorientierte Mathematik II

Abgabe: 02.06.2023 über den ComaJudge bis 17 Uhr

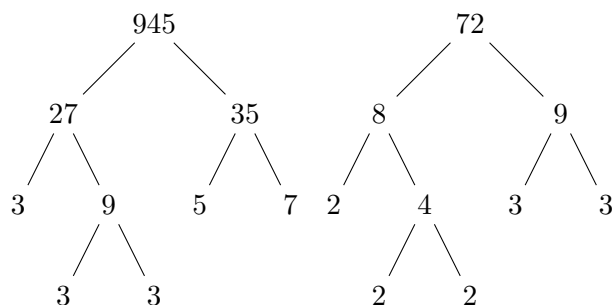
### Definition

Ein *Zerlegungsbaum*  $T(n)$  einer Zahl  $n$  ist ein binärer Baum, der rekursiv definiert ist durch

1. Eine Primzahl ist ein Zerlegungsbaum mit einem Blatt.
2. Für  $n > 0$  wird der Zerlegungsbaum durch die linken und rechten Teilgraphen  $T(a)$ ,  $T(b)$  definiert. Dabei sind  $a, b \in \mathbb{N}$  so gewählt, dass  $n = ab$  mit  $a \leq b$  und  $b - a$  minimal.

Zwei Zahlen haben die gleiche *Zerlegungsstruktur*, wenn sie die gleiche Baumstruktur haben. Zum Beispiel haben die Zahlen 945 und 72 die gleiche Zerlegungsstruktur.

### Beispiel



### Aufgabenstellung

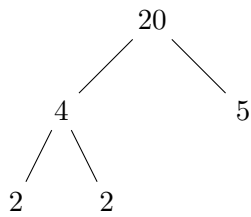
Implementieren Sie die Definition in Julia. Schreiben Sie dazu einen Typ `FactorTree` mit folgenden Eigenschaften

- `value::Int`: Die zu Faktorisierende Zahl.
- `left::Union{Int,FactorTree}`: Der linke Teilgraph.
- `right::Union{Int,FactorTree}`: Der rechte Teilgraph

Zusätzlich benötigen Sie einen Konstruktor `FactorTree(v::Int)::FactorTree` für diesen Typ und die folgenden Funktionen.

`getFactors(t::FactorTree)::Dict{Int, Int}`: Gibt die Primfaktorzerlegung der Wurzel des Zerlegungsbaums als Dictionary zurück, wobei ein `key` des Dictionarys die Primzahl und der `value` der Exponent der Primzahl ist.

`getShape(t::FactorTree)::String`: Gibt die Struktur des Zerlegungsbaums in Form eines Strings zurück. Die Kodierung ist frei wählbar. Ein Beispiel für 20 wäre `"f(p2|p)"`, was den Baumstruktur vollständig als Produkt `f(*|*)` eines Primzahlprodukts `p2` und einer weiteren Primzahl `p` beschreibt.



`compareShape(t::FactorTree, h::FactorTree)::Boolean`: Gibt an, ob zwei Zerlegungsbäume die gleiche Struktur haben oder nicht.

`computeShapes(n::Int)::Dict{String, Vector{Int}}`: Liefert ein Dictionary aller Zerlegungsstrukturen von Zahlen kleiner als eine positive ganze Zahl  $n$ . Die **keys** sind dabei die Kodierung einer Zerlegungsstruktur und die **values** ein Vektor von ganzen Zahlen mit genau dieser Zerlegungsstruktur.

*# Beispiel*

```

t = FactorTree(20);
getFactors(t)
>Dict{Int64, Int64} with 2 entries:
>  5 => 1
>  2 => 2
getShape(t)
> "f(p2|p)"
t2 = FactorTree(945);
t3 = FactorTree(72);
compareShape(t2, t3)
> true
computeShapes(10);
>Dict{String, Vector{Int64}} with 3 entries:
>  "p2"      => [4, 6, 9, 10]
>  "f(p|p2)" => [8]
>  "p"       => [1, 2, 3, 5, 7]
  
```