Technische Universität Berlin **SoSe 2023**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Antje Schulz
Prof. Dr. Michael Joswig
Dr. Frank Lutz, Martin Knaack, Marcel Wack

6. Programmieraufgabe Computerorientierte Mathematik II

Abgabe: 09.06.2023 über den ComaJudge bis 19 Uhr

Aufgabe

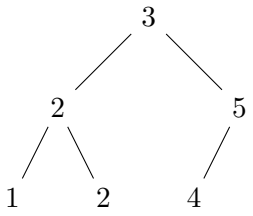
Implementieren Sie die Suchbaum Datenstruktur aus der Vorlesung. Dazu benötigen Sie ein `Type Node` mit den folgenden Eigenschaften

- `key::Int` Den Key des Knoten im Baum.
- `left::Union{Node,Nothing}` Der linke Teilbaum des Knotens.
- `right::Union{Node,Nothing}` Der rechte Teilbaum des Knotens.
- `parent::Union{Node,Nothing}` Der vorherige Knoten.

Implementieren Sie die folgenden Funktionen

- `getKeyList(tree::Node)::Vector{Int}` Liefert die Liste der Keys in aufsteigender Reihenfolge.
- `find(tree::Node,k::Int)::Union{Node,Nothing}` Gibt den `Node` mit dem Schlüssel `k` zurück, wenn er existiert. Andernfalls wird `nothing` zurückgegeben.
- `min(tree::Node)::Int` Gibt den minimalen Key im Baum zurück.

Ein Baum wird durch eine *pre-order*-Reihenfolge in einer Zeichenfolge kodiert. Konkret bedeutet dies, dass zuerst der Knoten selbst, dann sein linkes Blatt und dann sein rechtes Blatt angegeben werden. Als Beispiel gibt `4(1,5)` einen Baum mit der Wurzel 4, dem linken Blatt 1 und dem rechten Blatt 5 an. Hier ist der Baum für die Zeichenfolge `3(2(1,2),5(4,))`.



Schreiben Sie folgende Funktion

- `fromString(str::String)::Node` Konvertiert einen als String kodierten Baum in den `Type Node` um und prüft, ob es sich tatsächlich um einen Suchbaum handelt. Wenn nicht, wird zusätzlich zum Output eine Warnung `Der Baum ist kein Suchbaum!` zurückgegeben.