TU Ilmenau, Fakultät IA
FG Telematik/Rechnernetze
Prof. Dr.-Ing. G. Schäfer
http://www.tu-ilmenau.de/telematik/pua

TECHNISCHE UNIVERSITÄT
ILMENAU

Programmierung und Algorithmen WS 23/24
Übungsblatt 5

Die Lösungen der Aufgaben sind bis zum 19.11.23, 23:59 Uhr abzugeben.
Die Besprechung der Aufgaben erfolgt in KW 47.

Aufgabe 1 (Emulator für Markov-Algorithmen) — 5 Punkte

Implementieren Sie in Java einen Emulator für Markov-Algorithmen, die als Markov-Tafel notiert sind (siehe Kapitel 5, Folie 40). Der Emulator soll bei jeder Anwendung einer Regel den vorherigen Text, den Text nach Anwendung der Regel sowie die Zeilennummer der angewendeten Regel ausgeben.
Der Markov-Algorithmus selbst soll als zweidimensionales `String`-Array dargestellt werden. Die erste Dimension entspricht den Regeln/Zeilen des Algorithmus. Die Zeilennummer ist dabei implizit durch den Index im Array gegeben. Jede Regel besteht wiederum aus einem `String`-Array[1] mit vier Einträgen: Die linke Seite der Regel (was soll ersetzt werden), die rechte Seite der Regel (was soll stattdessen eingesetzt werden), die nächste Zeilennummer falls die Regel angewendet wurde und die nächste Zeilennummer falls die Regel nicht angewendet wurde. Das leere Wort ε soll durch einen leeren `String` dargestellt werden.
Das Beispiel aus Kapitel 5, Folie 46 kann dementsprechend wie folgt kodiert werden:

```
String[][] table = {{"|",  "#|",  "1", "3"},
                    {"#|", "||#", "1", "2"},
                    {"#",  "",    "3", "-1"}};
```

Die *Eingabe* für den Markov-Algorithmus können Sie entweder über die Kommandozeilenparameter Ihres Programms oder über unsere `IOUtils`-Bibliothek realisieren.

Hinweis: Das Einlesen der Markov-Tafel ist nicht Teil der Aufgabe, Sie dürfen die Tafel also wie im obigen Beispiel direkt in Ihr Java-Programm kodieren.

Hinweis: Die Referenz[2] der `String`-Klasse könnte hilfreich sein. Falls Sie ferner zur Implementierung Java's reguläre Ausdrücke verwenden möchten (viele Methoden der `String`-Klasse arbeiten damit), bedenken Sie, dass die Regeln des Markov-Algorithmus Zeichen enthalten könnten, welche in einem regulären Ausdruck eine besondere Bedeutung haben[3].

Aufgabe 2 (Registermaschinen) — 5 Punkte

Die *Catalan-Zahlen* C_k können durch folgende Formel berechnet werden:

$$C_0 = 1 \qquad C_{k+1} = \frac{(4k+2)C_k}{k+2}$$

Entwickeln Sie ein Registermaschinenprogramm, welches $n \in \mathbb{N}_0$ als Eingabe im ersten Speicherregister erhält und die Catalan-Zahl C_n als Ausgabe im zweiten Speicherregister berechnet.

Hinweis: Alle Catalan-Zahlen sind natürliche Zahlen. Damit die Berechnung auf der Registermaschine das korrekte Ergebnis liefert, müssen die Rechenoperationen in der richtigen Reihenfolge ausgeführt werden.

[1] Arrays können keine „gemischten" Datentypen enthalten, deshalb müssen die Zeilennummern ebenfalls als `String` kodiert werden. Zur Umwandlung in einen `int` kann die Methode `Integer.parseInt(String s)` verwendet werden
[2] https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html
[3] https://stackoverflow.com/questions/60160/how-to-escape-text-for-regular-expression-in-java

Aufgabe 3 (Markov-Algorithmen) **3 + 2 Punkte**

(a) Geben Sie einen Algorithmus als *Markov-Regeln* (siehe Kapitel 5, Folie 39; hier bitte keine Markov-Tafel verwenden!) an, der die Zeichen einer gegebenen Zeichenkette über dem Alphabet $\{a, b, c, d\}$ aufsteigend sortiert. Beispiel: $cabbacdaa \rightarrow aaaabbccd$.

(b) Demonstrieren Sie Ihren Algorithmus durch vollständige Ausführung auf der Eingabe *cabbacdaa*. Geben Sie dazu alle Zwischenschritte an und unterstreichen Sie jeweils den Teil der Zeichenkette, der im folgenden Schritt ersetzt wird.

Aufgabe 4 (Markov-Algorithmen) **4 + 2 Punkte**

(a) Geben Sie einen Algorithmus als *Markov-Tafel* an, der eine gegebene Zahl in Binärdarstellung in ihre Unärdarstellung umrechnet. So sollte etwa bei der Eingabe `101` die Ausgabe Ihres Algorithmus `|||||` sein.

(b) Demonstrieren Sie Ihren Algorithmus durch vollständige Ausführung auf der Eingabe `101`. Geben Sie dazu alle Zwischenschritte an und unterstreichen Sie jeweils den Teil der Zeichenkette, der im folgenden Schritt ersetzt wird.

Aufgabe 5 (Abstrakte Maschinen) **5 + 2 Punkte**

Sei $T(n)$ die Summe aller Teiler einer Zahl $n \in \mathbb{N}^+$, ausgenommen der Zahl selbst. Eine Zahl n heißt *perfekt* bzw. *vollkommen*, wenn $T(n) = n$ gilt. Beispiele sind:

$$T(6) = 3 + 2 + 1 = 6$$

$$T(28) = 14 + 7 + 4 + 2 + 1 = 28$$

(a) Definieren Sie eine abstrakte Maschine $M = (X, Y, K, \alpha, \omega, \tau, \sigma)$ welche bestimmt, ob eine gegebene Zahl eine perfekte Zahl ist. Die Ein- und Ausgabemengen sind dementsprechend wie folgt definiert:

- $X = \mathbb{N}_0$
- $Y = \texttt{Bool} = \{\texttt{true}, \texttt{false}\}$

Zu definieren ist noch eine sinnvolle Menge von Konfigurationen K sowie die Eingabefunktion α, die Ausgabefunktion ω, die Transitionsfunktion τ und die Stoppfunktion σ.

(b) Zeigen Sie die Funktionsweise der Maschine durch vollständige Ausführung mit der Eingabe 6. Geben Sie dafür die Folge der durchlaufenen Konfigurationen aus K an.

Hinweis: Sie dürfen die Modulo-Operation auf natürlichen Zahlen verwenden. Für Fallunterscheidungen können Sie die Syntax wie im Beispiel auf Folie 32, Kapitel 5 verwenden.