

Restsprache

Definition

Für eine gegebene Sprache $A \subseteq \Sigma^*$ und ein beliebiges Wort $u \in \Sigma^*$ definieren wir A/u , die “Restsprache bzgl. u ” folgendermaßen:

$$A/u = \{w \in \Sigma^* \mid wu \in A\}$$

In anderen Worten: A/u ist die Menge aller Wörter, die wir erhalten, wenn wir alle Wörter aus A löschen, die nicht auf u enden und dann von den verbleibenden jeweils das u am Ende entfernen.

Beispiel: Seien $A = \{\varepsilon, a, aa, ba\}$ und $B = \{a, b\}^* \cup \{b, c\}^*$. Dann gilt:

- $A/a = \{\varepsilon, a, b\}$, $A/b = \emptyset$
- $B/a = \{a\}^*$, $B/b = B$

Aufgabenstellung

Diese Aufgabe besteht aus zwei Teilen, die getrennt abgegeben und bepunktet werden. Dieses Blatt behandelt Teil C, welcher wie folgt lautet:

1. Überlegen Sie sich auf Papier, wie eine rekursive Prozedur f aussehen könnte, die für einen gegebenen regulären Ausdruck $r \in \text{RE}(\Sigma)$ und ein gegebenes Zeichen $a \in \Sigma$ einen regulären Ausdruck $f(r, a)$ berechnet, sodass $L(f(r, a)) = L(r)/a$ ist.
2. Nur für Java-Nutzer: Machen Sie sich mit dem Visitor-Pattern für reguläre Ausdrücke vertraut, das in dem Interface `Regex.Visitor` realisiert ist. Beispiele für dessen Verwendung finden Sie in den anderen Instanzen für `Regex.Visitor` im Template (z.B. `Regex.EvenOdd`).
3. Vervollständigen Sie die Implementierung der Klasse `RegexRemainder`, die eine Instanz des Visitor-Patterns darstellt.

Eingabe

Da es bei einigen von Ihnen auf dem letzten Blatt Probleme mit Unicode-Zeichen gab, wurde die Syntax für reguläre Ausdrücke so angepasst, dass keine Unicode-Zeichen mehr verwendet werden. Die regulären Ausdrücke \emptyset bzw. ε werden als `{ }` bzw. `()` eingegeben.

Beliebig viele Testfälle (einer pro Zeile), gefolgt von der Zeile `END`.

Jeder Testfall besteht aus einem Zeichen a und einem regulären Ausdruck r , getrennt durch ein Semikolon `;`.

Ausgabe

Für jeden Testfall einen regulären Ausdruck r' , sodass $L(r') = L(r)/a$.

Beispiele

Sample Input 1

```
a; {}  
a; ()  
a; a  
b; a  
a; aa  
a; ab  
a; ba  
a; a|b  
b; a|b  
a; b(a|())  
a; a*  
a; ba*  
a; a*b*  
a; b*a*  
a; (b|())*a*  
b; (b|())*b*  
a; a*(b|())*  
b; b*(b|())*  
END
```

Sample Output 1

```
{ }  
{ }  
( )  
{ }  
a  
{ }  
b  
( )  
( )  
b  
a*  
ba*  
a*  
b*a*  
( ) | b ) * a *  
( ) | b ) * b * | ( ) | b ) *  
a *  
b * ( ) | b ) * | b *
```