Algorithmen

Entwerfen Sie für folgende Aufgaben Algorithmen in Form von Ablaufdiagrammen. Verwenden Sie dazu die in der Vorlesung und Übung gezeigte Notation. Testen Sie Ihre Algorithmen mittels Schreibtischtests für einige interessante Eingaben.

Aufgabe 1: Zollgebühr (6 P.)

Der Algorithmus `customs(↓price, ↓amount, ↑fee)` berechnet für `amount` Stücke eines Produktes (jedes mit dem Wert `price`) wie viel Zoll bei der Einfuhr dafür bezahlt werden muss. Berechnen Sie dazu den Gesamtpreis aller Produkte. Bei einem Gesamtpreis bis zu 100 muss kein Zoll gezahlt werden. Ansonsten muss für den Betrag, der 100 übersteigt, 20% Zoll gezahlt werden.

Hinweis: Sie können davon ausgehen, dass `price` und `amount` positiv sind.

Tabelle 1: Beispiele für `customs`.

↓price	↓amount	↑fee
50	1	0
50	2	0
50	3	10
1	101	0.2
102	1	0.4

Aufgabe 2: Größte Ziffer in Zahl (6 P.)

Der Algorithmus `maxDigit(↓number, ↑max)` liefert zu einer Zahl (`number`) deren größte Ziffer (`max`). Machen Sie sich dabei zu Nutze, dass bei einer Ganzzahldivision durch 10 die Einerstelle einer Zahl "abgeschnitten wird", beispielsweise 137 / 10 = 13. Den Rest einer Ganzzahldivision können Sie über die Modulo-Operation (%) abfragen, beispielsweise 137 % 10 = 7 ($137 = 13 * 10 + 7$ Rest).

Hinweis: Sie brauchen keine negativen Zahlen berücksichtigen.

Tabelle 2: Beispiele für `maxDigit`.

↓number	↑max
0	0
1	1
1243	4
1500	5
9876	9

Aufgabe 3: Exponentialrechnung (6 P.)

Der Algorithmus `power(↓base, ↓exponent, ↑result)` soll als `result` den Wert $base^{exponent}$ liefern. Bei einem `exponent` von 0 wird als `result` also 1 zurückgeliefert, für positive Exponenten muss `base` `exponent`-mal zu `result` multipliziert werden. Zum Beispiel ist $base^3$ der Wert $1 \cdot base \cdot base \cdot base$.

Hinweis: Sie können davon ausgehen, dass `exponent` eine ganze Zahl ≥ 0 ist. Nutzen Sie eine Schleife, um `result` stückweise zu berechnen (bitte wie in der Angabe oben gegeben und nicht wie in der Vorlesung gezeigt).

Tabelle 3: Beispiele für `power`.

↓`base`	↓`exponent`	↑`result`
2	0	1
10	0	1
2	3	8 $(1 \cdot 2 \cdot 2 \cdot 2)$
10	4	10000 $(1 \cdot 10 \cdot 10 \cdot 10 \cdot 10)$
-1	3	-1 $(1 \cdot (-1) \cdot (-1) \cdot (-1))$

EBNF

Definieren Sie für folgende Aufgabe EBNF Grammatikregeln, um die im Text beschriebenen Datenstrukturen abzubilden.

Aufgabe 4: ISO 8601 (6 P.)

Gesucht sind Grammatikregeln für die Definition eines Zeitstempels nach ISO 8601, welcher aus Datum sowie optional der Uhrzeit sowie Zeitzoneninformation besteht. Das Datum ist verpflichtend und besteht aus dem Jahr (vierstellige Zahl), dem Monat (zweistellige Zahl) und dem Tag (zweistellige Zahl), jeweils getrennt durch einen Bindestrich. Die Uhrzeit ist optional und besteht aus Stunden (zweistellige Zahl), Minuten (zweistellige Zahl) und Sekunden (zweistellige Zahl), jeweils durch Doppelpunkt getrennt. Die Zeit wird mittels dem Zeichen "`T`" vom Datum getrennt. Ist eine Zeitangabe vorhanden, kann danach optional eine Zeitzonenangabe folgen. Diese besteht entweder aus dem Zeichen "`Z`" (für die UTC-Zeitzone) oder aus einer expliziten Angabe der Zeitverschiebung in Stunden und Minuten (jeweils zweistellige Zahl getrennt durch Doppelpunkt) sowie einem Vorzeichen als Präfix ("`+`" oder "`-`").

Sie müssen nicht prüfen, ob die Zeitstempel tatsächlich gültig sind. Ungültige Datums- (`2020-42-34`) oder Zeitwerte (`88:63:90+01:39`) sind ebenfalls erlaubt. Beachten Sie weiters, dass eine Zonenangabe nur dann erfolgen kann, wenn auch eine Zeitangabe vorhanden ist.

Sie dürfen annehmen, dass es bereits eine Definition für `Digit` gibt, die die Ziffern 0 bis 9 abdeckt.

Beispiele für Zeitstempel:

- `2001-12-30T10:38:11+01:00`
- `2021-01-01T19:17:48-02:30`
- `1010-30-00T26:99:18-00:01`
- `2002-02-22T23:01:12Z`
- `2018-03-30T10:00:80`
- `1999-02-32`

Zusätzliche Anweisungen

- Achten Sie darauf, dass die Algorithmen auch bei Grenzfällen funktionieren. Testen Sie interessante Grenzfälle durch Schreibtischtests und legen Sie die Schreibtischtests ihrer Abgabe bei!
- Strukturieren Sie Ihre Grammatik mithilfe mehrerer Regeln. Ihre Grammatikregeln sollen **nur die Struktur** definieren, und **keine semantischen Überprüfungen** durchführen. Sie müssen also **nicht prüfen**, ob der Tag im jeweiligen Monat existiert oder die Uhrzeit in einem gültigen Bereich liegt.

Abzugeben

- Ablaufdiagramme (`Aufgabe<Nr>.pdf`, `Aufgabe<Nr>.jpg` oder `Aufgabe<Nr>.png`)
- Schreibtischtests (`Schreibtischtest<Nr>.pdf`, `Schreibtischtest<Nr>.jpg` oder `Schreibtischtest<Nr>.png`)
- EBNF Grammatikregeln (`Aufgabe04.pdf` oder `Aufgabe04.txt`)