Übungsblatt 1

Aufgabe 1 - Syntaxdiagramm

Erstelle ein Syntaxdiagramm, das den Aufbau eines *Bezeichners* definiert:
– Ein Bezeichner muss mindestens ein Zeichen lang sein.
– Ein Bezeichner muss mit einem *Buchstaben* oder dem Zeichen `_` beginnen.
– Ab der zweiten Position können beliebige *Buchstaben*, *Ziffern* oder das Zeichen `_` folgen.

Gehe davon aus, dass für die beiden Nichtterminalsymbole *Buchstabe* und *Ziffer* bereits Syntaxdiagramme vorliegen.

Aufgabe 2 - Syntaxdiagramm

Erstelle ein Syntaxdigramm, das den Aufbau von (vereinfachten) arithmetischen Ausdrücken definiert. Ein Ausdruck kann aus den beiden Operatoren + und *, aus *Buchstaben* und aus Klammerungen mit `(` ... `)` bestehen.
– Ein Ausdruck beginnt mit einem *Buchstaben* oder `(`.
– Vor einem Operator muss ein *Buchstabe* oder `)` stehen.
– Hinter einem Operator muss ein *Buchstabe* oder `(` stehen.
– Zwei *Buchstaben* dürfen nicht unmittelbar aufeinander folgen.
– Vor einem *Buchstaben* darf kein `)` stehen.
– Hinter einem *Buchstaben* darf kein `(` stehen.
– In keinem beliebigen Anfangsstück des Ausdrucks darf es weniger `(` als `)` geben.
– In einem Ausdruck muss es die gleiche Anzahl von `(` und `)` geben.
– Nicht erlaubt ist die Folge `()`.

Gehe davon aus, dass für das Nichtterminalsymbol *Buchstabe* bereits ein Syntaxdiagramm vorliegt.

Hinweis: Diese Regeln beschreiben letztlich das, was man unter einem Ausdruck mit Klammerung erwarten würde.

```
(a)     (B)+(c*(x+e))          u+v+w          u+v*w
```

Aufgabe 3 - Algorithmen konzipieren

Skizziere Algorithmen in Umgangssprache für die folgenden Abläufe. Überlege insbesondere, wie während der Ausführung deren Ende feststellt werden kann. Gehe bei allen Algorithmen davon aus, dass vor Dir zwei unsortierte Stapel mit je etwa 500 Klausuren liegen, auf deren Deckblatt die Matrikelnummer des bearbeitenden Studierenden steht.

- Gibt es mindestens einen Studierenden, der beide Klausuren mitgeschrieben hat?
- Gibt es keinen Studierenden, der beide Klausuren mitgeschrieben hat?
- Wie viele Studierende haben beide Klausuren mitgeschrieben?
- Gibt es im Stapel 1 mindestens eine Matrikelnummer, die größer ist als jede Matrikelnummer im Stapel 2?
- Ist im Stapel 1 jede Matrikelnummer größer als jede Matrikelnummer im Stapel 2?

Was ändert sich an den Algorithmen, wenn die beiden Klausurenstapel aufsteigend nach Matrikelnummern sortiert sind?

Aufgabe 4 - Ganzzahlige Berechnungen

Benutze in den Rümpfen der Methoden nur die `return`-Anweisung und die mathematischen Operatoren `+`, `-`, `/` oder `*` (Multiplikation). Beachte, dass `/` für Werte des Typs `int` die *ganzzahlige* Division durchführt.

Implementiere die folgenden Methoden. Gehe davon aus, dass nur positive Werte als Argumente übergeben werden.

- Die Methode `int remainder( int dividend, int divisor )` soll den Wert zurückgeben, der als Rest der Division von `dividend` durch `divisor` bleibt.
- Die Methode `int isOdd( int value )` soll `1` zurückgeben, falls dem Parameter `value` ein ungerader Wert als Argument übergeben wird. Sie soll `0` zurückgeben, falls dem Parameter `value` ein gerader Wert übergeben wird.
- Die Methode `int isEven( int value )` soll `1` zurückgeben, falls dem Parameter `value` ein gerader Wert als Argument übergeben wird. Sie soll `0` zurückgeben, falls dem Parameter `value` ein ungerader Wert übergeben wird.
- Die Methode `int toEven( int value )` soll Folgendes leisten: Falls dem Parameter `value` ein ungerader Wert als Argument übergeben wird, soll der nächstgrößere gerade Wert zurückgegeben werden. Falls dem Parameter `value` ein gerader Wert übergeben wird, soll dieser (unverändert) zurückgegeben werden.
- Die Methode `int isDivisible( int dividend, int divisor1, int divisor2 )` soll `0` zurückgeben, falls `dividend` sowohl durch `divisor1` als auch durch `divisor2` ganzzahlig – also ohne Rest – teilbar ist. Sonst soll ein beliebiger Wert ungleich `0` zurückgegeben werden.