Aufgabe 1: *Struct Rational*

Viele Operationen mit rationalen Zahlen erfordern neben der Verwendung des ***ggT*** (vergleiche Aufgabe 3 Aufgabenblatt 1) auch die Bestimmung des kleinsten gemeinsamen Vielfachen (***kgV***).

a) Schreiben Sie eine Funktion, die das ***kgV*** von zwei ganzen Zahlen bestimmt.

b) Erstellen Sie ein ***struct Bruch***, der eine rationale Zahl mit Zähler und Nenner repräsentiert. Deklarieren Sie diese Struktur anschließend mittels ***typedef*** als neuen Datentyp ***Rational***.

c) Definieren Sie zu dem neuen Datentyp folgende Funktionen:

- ```
  //Kürzt einen Bruch
  Rational kuerze(Rational r)
  ```
- ```
  //Berechnet die Summe von a und b
  Rational addiere(Rational a, Rational b)
  ```
- ```
  //Berechnet die Differenz von a und b
  Rational subtrahiere(Rational a, Rational b)
  ```
- ```
  //Berechnet das Produkt von a und b
  Rational multipliziere(Rational a, Rational b)
  ```
- ```
  //Berechnet den Quotient von a und b
  Rational dividiere(Rational a, Rational b)
  ```
- ```
  //Berechnet aus Zähler und Nenner eine Fließkommazahl
  float toFloat(Rational r)
  ```
- ```
  //Gibt einen Bruch auf der Kommandozeile aus
  void ausgabe(Rational r)
  ```

d) Schreiben Sie ein Programm, das die in Aufgabe 2 entwickelten Funktionen nacheinander testet. Dazu muss Ihr Programm die Werte für Zähler und Nenner der rationalen Zahlen zur Laufzeit von der Tastatur einlesen. Verwenden Sie dazu die Funktion ***scanf()*** aus ***stdio.h***.

Aufgabe 2: *Struct/Union Color*

In dieser Aufgabe soll einen neuen Datentyp zur Beschreibung der Farbwerte eines Pixels definiert werden. Der Datentyp soll die Farbwerte sowohl als RGB-Angaben als auch als HEX-Werte speichern können.

a) Implementieren Sie einen `struct`, der die Farbwerte R, G und B enthält. Die Werte sind im Bereich 0 bis 255.

b) Definieren Sie einen neuen Datentyp, der diesen `struct` zusätzlich als vorzeichenloser Integer darstellen kann. Verwenden Sie hierfür den Datentyp `union`.

c) Testen Sie den neuen Datentyp in einem vollständigen C-Programm.
  ```

Aufgabe 3: *Struct Messung*

In einer Werkhalle sollen Messwerte von Produkttests gespeichert werden. Im Folgenden sind die Messwerte und deren Wertebereiche festgelegt:

Spannung:	-250 V bis +250 V
Stromstaerke:	-10.0 A bis +10.0 A
Druck:	0 Bar bis 20 Bar

a) Erstellen Sie einen ***struct Messung***, der die oben dargestellten Messwerte speichern kann. Überlegen Sie sich welche Datentypen für die Messwerte geeignet wären.

b) Erweitern Sie die Struktur ***struct Messung*** um einen Zeitstempel. Dazu müssen Sie zunächst die Struktur um eine Komponente vom Typ ***struct tm*** erweitern. Wählen Sie den Namen ***zeit*** für diese Komponente. Der ***struct tm*** ist Teil der C-Bibliothek und kann nach dem Einbinden des Header ***time.h (#include <time.h>)*** verwendet werden.

c) Testen Sie ***struct Messung***, indem Sie in ***main()*** eine Variable dieses Typs erstellen und diese mit Werten befüllen. Lesen Sie die Werte für Spannung, Strom und Druck mit der Funktion ***scanf()*** von der Tastatur ein. Die Funktion ***time()*** liefert die aktuelle Zeit in einer Variablen vom Typ ***time_t***. Mit der Funktion ***localtime()*** kann diese in den Typ ***struct tm*** konvertiert werden. Die Funktionen und Typen sind Bestandteil der C-Bibliothek und lassen sich nach dem Einbinden des Header ***time.h (#include <time.h>)*** verwenden.

Beispiel:

```
time_t t;
time(&t);
struct tm zeit = *localtime(&t);
```

d) Schreiben Sie eine Ausgabefunktion ***ausgabe()***, die den Inhalt einer Variablen vom Typ ***struct Messung*** in einer lesbaren Form auf der Kommandozeile ausgibt.