Thema, Ziele: Testen allgemein

Aufgabe 1: Rabatt Rechnung Äquivalenzklassen

Ein Supermarkt ist an den Wochentagen (Mo.-Sa.) von 08:00 bis 20:00 Uhr geöffnet. Ein Kassenprogramm in dem Supermarkt hat folgende Anforderungen:

1. Ab einem Rechnungsbetrag von insgesamt CHF 150 wird ein Rabatt von CHF 10 gewährt.
2. Da es morgens immer ziemlich leer im Supermarkt ist, soll ein weiterer Rabatt für grösseren Umsatz sorgen: Auf alle Waren, die morgens bis 10:00 Uhr eingekauft werden, wird ein Rabatt von 5% gewährt (Frühkaufrabatt).
3. Der Frühkaufrabatt kann nicht mit dem Rechnungsbetrag Rabatt kombiniert werden .Es soll der für den Kunden günstigere der beiden Rabatte gelten.

Der für den Rabatt entscheidende Baustein des Kassenprogramms sei die Funktion:

```
size_t rabatt(size_t rechnungsbetrag, size_t stunde, size_t minute)
```

Der zurückgegebene Rabatt und der Rechnungsbetrag werden als Rappen-Beträge angesehen, weil es keine kleinere Einheit gibt. Der Datentyp `size_t` wird gewählt, weil mit ihm die grösstmöglichen `unsigned`-Zahlen darstellbar sind. Eine mögliche Ursache für Fehler, die durch eine Typumwandlung entstehen, entfällt dadurch. Aus Plausibilitätsgründen wird eine Obergrenze von CHF 10‘000 festgelegt. Ein höherer Wert gilt als Fehler. Pfandgeld wird nicht berücksichtigt, deshalb muss der Rechnungsbetrag > CHF 0 sein.

Definieren Sie alle Äquivalenzklassen (gültig und ungültig).

Aufgabe 2: Rabatt Rechnung Testfälle mit Grenzwertanalyse

Definieren Sie konkrete Testfälle (Input und Output) für die in Aufgabe 1 gebildeten Äquivalenzklassen mittels Grenzwertanalyse.

Falls ungültigen Äquivalenzklassen wird eine Exception von der Funktion `rabatt` geworfen.

Hinweis: Das Makro `SIZE_MAX` aus dem Header *stdint.h* repräsentiert der maximale Wert welcher mit `size_t` dargestellt werden kann.

Aufgabe 3: Testfälle für *VoltageDivider* Anwendung

Definieren Sie alle Testfälle für die Funktionen `VoltageDivider::calc` (siehe Vorlage *VoltageDivider.h*). Verwenden Sie dazu Äquivalenzklassen und die Grenzwertanalyse. Die resultierenden Testfälle dienen als Vorlage für das nächste Praktikum, damit automatisiert getestet werden kann.

Hinweise:

- Das Makro `DBL_MAX` aus dem Header *cfloat* repräsentiert der maximale Wert welcher mit `double` dargestellt werden kann.
- Die Funktion `nextafter` gibt der nächste Repräsentant eines double Wertes zurück. Diese Funktion ist im *cmath* Header.