Programmiersprachen 2

Prof. Dr. Detlef Krömker
Alexander Wolodkin

..

Übungsblatt Nr. PS2.1-V0.9

1
Ausgabe: 27.11.2017
Abgabe: **Freitag,** 22.12.2017 - 16:00 Uhr

___ / 50 Punkten

Software Testing

Hinweis: Dieses Aufgabenblatt ist mit einer Bearbeitungszeit von rund 4 Wochen dafür ausgelegt von einer/m Studierenden bearbeitet zu werden. Neben der Implementierung wird bei der Bearbeitung Wert auf die folgenden Punkte gelegt: Dokumentation

- Strukturierung und Einhalten des Style-Guides
- Entwicklung der Testfälle

.
Achtung: Achten Sie darauf die Variable __author__ in allen Quellcode Dateien korrekt zu setzen. Abgaben die nicht dieser Vorgabe entsprechen werden nicht bewertet! Quellcode muss als .py Datei und alles Weitere als .pdf Datei abgegeben werden. Bei Abgaben mehrerer Dateien müssen diese als .zip zusammengefasst werden. Abgaben, die nicht diesen Regeln entsprechen, werden ebenfalls nicht bewertet! Außerdem muss der Name der/des Bearbeiter*in in jeder abgegebenen .pdf Datei zu finden sein.

Um den Leistungsschein in PS 2 zu erwerben, benötigen Sie mindestens 50% der maximal erreichbaren Punkte in den PS2-Hausübungen.

Aufgabe 1.1 (8 Punkte) Die Programmieraufgabe ist relativ klein und einfach:

Implementieren Sie eine Funktion `dateconvert(date),` die ein Datum(date) (nach dem gregorianischen Kalender) im Zeitraum 1. Januar 1801 bis 31. Dezember 2200 im Format

YYYYMMDD

YYYY ist die 4-stellige Jahreszahl,
MM ist die zweistellige Monatszahl in dem Jahr YYYY
DD ist die zweistellige Tageszahl in dem Monat DD

übergeben bekommt. Die Funktion dateconvert (date) soll als Tupel zurückliefern:

```
(german_date, british_date, american date, unix_date)
```

german_date, british_date und american_date sind Strings, unix_date ist ein integer

Die Datum-Strings sollen für den Funktionsparameter `19820807` folgendermaßen aussehen:

german_date: `"Sonnabend, 7. August 1982"`
british_date: `"Saturday, 7 August 1982"`
american date: `"Saturday, 08/07/1982"`

Für die Unix time nehmen Sie bitte 12.00 Uhr mittags an (noon). Die `unix_time` hat dann für dieses Beispiel den Wert `397562400.`

Überprüfen Sie unbedingt, ob die Eingabe ein gültiges Datum ist! Beachten Sie die Schaltjahre. Ggf. recherchieren Sie die Zusammenhänge und erzeugen Sie ggf. aussagefähige Fehlermeldungen.

Wie Sie wissen, dient ein solcher Test auch immer dem Qualitätscheck der Spezifikation – ggf. ergänzen Sie die Spezifikation. Geben Sie die Ergänzungen explizit an!.

Aufgabe 1.2 (4 Punkte) Entwickeln Sie unter Nutzung Ihrer Entwicklung aus Aufgabe 1.1 hieraus ein Programm mit Konsolen Ein- und Ausgabe. Die Eingabe des Datums an der Konsole darf entweder in dem Format des o.g. Funktionsparameters oder in der amerikanischen Schreibweise, also z.B. `"08/07/1982"` erfolgen. (Nach einem zu wählenden Prompt). Geben Sie alle vier genannten Datumsangaben in jeweils einer Zeile auf der Konsole aus.

Aufgabe 1.3
Ihre Aufgabe ist es, einen Komponententest (Blackbox Test) für die in 1.1 und 1.2 spezifizierten Programmteile zu entwickeln. Wie Sie wissen, dient ein solcher Test auch immer dem Qualitätscheck der Spezifikation – ggf. ergänzen Sie die Spezifikation. Geben Sie die Ergänzungen explizit an!.

Aufgabe 1.3a) (12 Punkte, je 6 für Funktion und Programm) Entwickeln Sie geeignete **Äquivalenzklassen** für die in 1.1 spezifizierte Funktion **und** das in 1.2 spezifizierte Programm. Wählen Sie auch geeignete Repräsentanten (= Testcases) für diese Äquivalenzklassen.

Für die Angabe der Äquivalenzklassen nutzen Sie bitte das beiliegende Word-File, dass Sie für die Abgabe natürlich ausgefüllt in ein .pdf wandeln. Es sind „natürlich“ mehr als 4 Testfälle anzugeben … Sie können die Tabelle aber einfach erweitern.

Versuchen Sie „**Äquivalenzklassenüberdeckung“** zu erreichen. Denken Sie insbesondere auch an die Negativ-Testfälle.

Aufgabe 1.3b) (14 Punkte, je 7 für Funktion und Programm) (Führen Sie wieder für die Funktion `dateconvert(date)` und das Programm eine **Grenzwertanalyse** für die identifizierten Äquivlenzklassen durch. Welche Testfälle kommen hinzu. Nutzen sie auch wieder die Word-Vorlage.

Falls Sie mehr als 20 zusätzliche Testfälle identifiziert haben, benutzen sie bitte einen Klassifikationsbaum zusammen mit der **Twowise-**Methode.

Aufgabe 1.4) (12 Punkte) Entwickeln Sie für die in 1.3a und b entwickelten Testfälle entsprechende Doctests.

Ein Tipp: Es ist gerade bei dieser Aufgabe durchaus sinnvoll einen Test-first Ansatz zu wählen. Ich möchte Sie nicht dazu zwingen, aber probieren Sie es doch einfach einmal aus.