Hinweise zu Organisation und Ablauf des Übungsteil der LVA 185.A03 Funktionale Programmierung im WS 2020/21

(Stand: 04.08.2020)

Online-Programmiermöglichkeit im Labor "Argentinierstr."

ACHTUNG: *Aufgrund von COVID-19-Beschränkungen ist das Labor voraussichtlich nicht in Präsenz nutzbar und der Zugang zum Übungsserver deshalb nur online (über ssh/scp-Verbindungen) möglich!*

Zur Bearbeitung, Lösung und Abgabe der Programmieraufgaben stehen Ihnen (im Rahmen der Kapazitäten) die Rechner im Labor im Erdgeschoss des Institutsgebäudes Argentinierstraße 8 zur Verfügung. Sollten die aktuellen COVID-19-Beschränkungen für das WS 2020/21 ausgesetzt werden und das Labor in Präsenz nutzbar sein, erreichen Sie das Labor über den kleinen Innenhof im Erdgeschoss. Einen Lageplan finden Sie auf:
`http://www.complang.tuwien.ac.at/ulrich/p-1851-E.html` .

Um die Aufgaben zu lösen, rufen Sie bitte den GHCi- oder Hugs-Interpretierer in der Kommandozeile einer Shell auf. Falls Sie die Übungsaufgaben auf Ihrem eigenen Rechner bearbeiten möchten, müssen Sie diese Interpretierer zunächst installieren. Sie sind frei verfügbar unter:

- Glasgow Haskell Compiler GHC und Interpretierer GHCi: `hackage.haskell.org/platform/`
- Hugs-Interpretierer: `www.haskell.org/hugs`

Der Aufruf der jeweiligen Interpretierervariante ist danach vom Betriebssystem abhängig.

Ausgabe von Angaben, Erst- und Zweitabgabe von Lösungen

Angaben werden regelmäßig freitags auf der Webseite
`http://www.complang.tuwien.ac.at/knoop/lehre/ws1920/fp185A03`
ausgegeben und können dort heruntergeladen werden.

Der Name von Abgabedateien ist stets auf dem Angabenblatt vorgegeben. Für jede Angabe ist der Abgabedateiname für Erst- und Zweitabgabe ident.

Die Erstabgabe erfolgt regelmäßig freitags (12:00 Uhr) eine Woche nach Angabenausgabe. Der Zweitabgabezeitpunkt richtet sich nach dem Beurteilungszeitpunkt der Erstabgabe (siehe Abschnitt "Abgabezeitpunkt von Zweitabgaben").

Die Abgabedatei muss für jede Angabe auf der Maschine `g0.complang.tuwien.ac.at` in Ihrem Home-Verzeichnis abgelegt werden, keinesfalls in einem Unterverzeichnis! Verwenden Sie ssh/scp oder vergleichbare Dienstprogramme, wenn Sie nicht unmittelbar auf der `g0` arbeiten, um auf der `g0` zu arbeiten und Dateien auf die `g0` zu kopieren.

Benachrichtigung über erreichte Punkte

Für die ersten vier Angaben sind je **50** Punkte erreichbar, für die letzten drei je **100** Punkte. Die Ergebnisse der Bewertung Ihrer abgegebenen Lösungen wird in einer Datei in Ihrem Home-Verzeichnis auf der `g0` abgelegt.

Abgabezeitpunkt für Zweitabgaben

Für jede Angabe können Sie nach Erhalt der Bewertung der Erstabgabe Verbesserungen vornehmen und als zweite Lösung abgeben. Der Zeitpunkt für diese zweite Abgabe richtet sich nach dem Zeitpunkt, an dem Sie die Bewertung der Erstabgabe erhalten.

Es ist der Freitag (12:00 Uhr), der auf den Mittwoch folgt, an dem Sie die Ergebnisse der Erstabgabe spätestens um 09:00 Uhr erhalten haben. Im Normalfall ist dies der Freitag zwei Wochen nach der Ausgabe einer Angabe.

Wichtig

• **Vorhalten von Aufgabenlösungen zu immer beiden Angabeabgabeterminen.** Zu beiden Abgabeterminen für eine Angabe und bis zum Erhalt der jeweiligen Bewertung (gleich ob Erst- oder Zweitabgabe für eine Angabe) müssen Sie Ihre Lösungen der Programmieraufgaben in einer entsprechend der Aufgabenvorgabe benannten Datei in Ihrem Home-Verzeichnis auf dem Übungsrechner `g0` zum automatischen Absammeln abgelegt bereit halten. Ihre Abgabedatei wird zu den genannten Zeitpunkten automatisch aus ihrem Home-Verzeichnis kopiert. Findet sich dort keine Abgabedatei, erhalten Sie für diesen Abgabetermin 0 Punkte. Ihre Abgabedatei darf deshalb **niemals** in einem Unterverzeichnis Ihres Home-Verzeichnisses stehen oder anders benannt sein als auf der Angabe vorgegeben.

• **Arbeit mit GHCi und Hugs; Bewertung ausschließlich unter GHCi.** Sie können Ihre Aufgabenlösungen sowohl mit Hugs als auch mit GHCi vorbereiten. Die *Bewertung* Ihrer Aufgabenlösungen erfolgt jedoch *ausschließlich unter GHCi in der auf dem Übungsrechner `g0` installierten Version.*

Wenn Sie Ihre Aufgabenlösungen nicht direkt auf der `g0` (im Labor "Argentinierstr." oder über eine `ssh`-Verbindung) unter GHCi entwickeln, müssen Sie sich deshalb nach erfolgter Übertragung Ihrer Abgabedatei auf die `g0` (z.B. über eine `scp`-Verbindung) *unbedingt vergewissern*, dass Ihre möglicherweise mit einem anderen Werkzeug entwickelte Lösung *auch unter GHCi auf dem Übungsrechner `g0` wie von Ihnen erwartet arbeitet.*

Machen Sie sich rechtzeitig mit der technischen Abwicklung von Aufgabenabgabe und Überprüfung des erwarteten Programmverhaltens auf der `g0` *vertraut, in jedem Fall rechtzeitig vor dem ersten Abgabetermin.*

Online-Tutorien zu Haskell, GHC/GHCi und Hugs

Online-Tutorien und -Manuale zu Haskell und zu Haskell-Interpretierern finden Sie z.B. unter `www.haskell.org/` und `www.haskell.org/tutorial/`; speziell für Hugs 98 unter `www.haskell.org/hugs/pages/hugsman/` ein *Online*-Manual. Lesen Sie besonders die ersten Abschnitte des Manuals, darunter Abschnitt 3 zum Thema „Hugs for beginners" und probieren Sie die darin beschriebenen Beispiele aus. Machen Sie sich mit den Haskell-Interpretierern GHCi und Hugs 98 mindestens so weit vertraut, dass Sie problemlos einfache Ausdrücke auswerten lassen können. Hinweise zur Arbeit mit Haskell (und mit Hugs 98, Kapitel 4) finden Sie auch in:

– H. Conrad Cunningham. *Notes on Functional Programming with Haskell.* Course Notes, University of Mississippi, 2007 (Chapter 4, Using the Hugs Interpreter).
`citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.2822&rep=rep1&type=pdf`

Fragen von allgemeinem Interesse zu Vortrag und Übung können Sie auch über das TISS-Forum zur Lehrveranstaltung zur Diskussion stellen.

Hinweise zu Haskell und möglichen (Anfangs-) Klippen und Hürden

Die Syntax von Haskell birgt im großen und ganzen keine besonderen Fallstricke und ist zumeist intuitiv, anfangs im Vergleich zu anderen Sprachen möglicherweise jedoch ungewohnt und deshalb "eingewöhnungsbedürftig".

Eine Hürde für neu mit Haskell beginnende Programmierer sind Einrückungen. Einrükkungen tragen in Haskell Bedeutung für die Bindungsbereiche und damit für die Programmbedeutung. Einrückungen müssen deshalb unbedingt eingehalten werden (vgl. z.B. Kap. 3.7 der Vorlesung zu „Funktions- und Programmlayout, Abseitsregel"). Alles, was zum selben Bindungsbereich gehört, muss in derselben Spalte beginnen. Diese in ähnlicher Form auch in anderen Sprachen wie `Occam` vorkommende Konvention erlaubt es, Strichpunkte und Klammern einzusparen.

Ein Anwendungsbeispiel in Haskell: Wenn eine Funktion mehrere Zeilen umfasst, muss alles, was nach dem „=" steht, in derselben Spalte beginnen oder noch weiter eingerückt sein als in der ersten Zeile. Anderenfalls liefern Übesetzer und Interpretierer dem Haskell-Programmierbeginner (scheinbar) unverständliche Fehlermeldungen wegen z.B. fehlender Strichpunkte.

Weiters sollen alle Funktionsdefinitionen und Typdeklarationen in Haskell-Programmen in derselben Spalte ganz links beginnen. Verwenden Sie keine Tabulatoren oder stellen Sie die Tabulatorgröße auf acht Zeichen ein.

Achten Sie auf richtige Klammerung, auch wenn Haskell vielfach keine Klammern verlangt, da sie aufgrund vordefinierter Prioritätsregeln automatisch ergänzt werden können. Im Zweifelsfall ist es gute Praxis, ggf. überflüssig zu klammern, um „Überraschungen" zu vermeiden. So entspricht zum Beispiel der Ausdruck „`potenz 2 -1`" dem Ausdruck „`(potenz 2) - (1)`" (oder kürzer: „`potenz 2 - 1`") und nicht dem Aufruf „`potenz 2 (-1)`", wie man möglicherweise erwarten könnte.

Operatoren haben immer eine niedrigere Priorität als das Hintereinanderschreiben von Ausdrücken (d.h. Funktionsanwendungen). Der Unterstrich „_" zählt zu den Kleinbuchstaben. Wenn Sie nicht sicher sind, verwenden Sie lieber mehr Klammern als aufgrund von Klammereinsparungsregeln möglicherweise nötig sind.

Beachten Sie, dass das Minussymbol kontextabhängig als Infix- (Subtraktion) oder Prefix-Operator (Vorzeichenwechsel) interpretiert wird.

Funktionsdefinitionen und Typdeklarationen können Sie nicht direkt im Haskell-Interpretierer schreiben, sondern nur aus Dateien laden. Genauere Hinweise zur Haskell-Syntax finden Sie z.B. unter `haskell.org/tutorial/` .