Human-Computer Interaction Group

Leibniz Universität Hannover

# Assignment 3

## Programmieren 1 – WiSe 25/26

**Prof. Dr. Michael Rohs, Falk Stock, M.Sc.**

Alle Assignments (bis auf das erste) müssen in Zweiergruppen bearbeitet werden. Ein Gruppenmitglied kann dabei die Lösung der Zweiergruppe gebündelt abgeben. Einzige Ausnahme ist dabei das erste Assignment, wo jedes Gruppenmitglied einzeln abgeben muss. Namen beider Gruppenmitglieder müssen sowohl in der PDF-Abgabe, als auch als Kommentar in jeglichen Quelltextabgaben genannt werden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 13.11. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/WiSe2024/Prog1. Die Abgabe muss aus einer einzelnen Zip-Datei bestehen, die den Quellcode, eine PDF für Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen auf.

Zum Bestehen der Studienleistung müssen Sie mindestens zwei von vier Punkten pro Assignment erzielen. Möchten Sie zusätzlich den Klausurbonus erreichen, müssen Sie über alle Assignments hinweg 75% der Punkte erreichen.

## Aufgabe 1: Skript (1 Punkt)

Das Skript unter https://postfix.hci.uni-hannover.de/files/prog1script-postfix/ beschreibt verschiedene Vorgehensweisen bei der Lösung von Programmieraufgaben.

a) Lesen Sie Kapitel 6 (Recipe for Itemizations) und beantworten Sie folgende Frage in eigenen Worten: Wieso wird in den Testfällen test~= statt test= verwendet?

b) Lesen Sie Kapitel 7 (Recipe for Compound Data) und beantworten Sie folgende Frage in eigenen Worten: Was ist die Aufgabe von constructor-Funktionen und was die von accessor-Funktionen?

c) Lesen Sie Kapitel 8 (Recipe for Variant Data) und beantworten Sie folgende Frage in eigenen Worten: Welchen Vorteil bietet datadef im Zusammenhang mit Variant Data?

d) Was war Ihnen beim Lesen der Kapitel 6 bis 8 unklar? Wenn nichts unklar war, welcher Aspekt war für Sie am interessantesten?

## Aufgabe 2: Tempomat (1 Punkt)

Ein Tempomat hat die Aufgabe, ein gegebenes Tempo möglichst genau zu halten. Entwickeln Sie eine Funktion zur Regelung eines Tempomats, welches abhängig vom aktuellen Tempo und dem Zieltempo entweder bremst, nichts tut, beschleunigt oder eine Notbremsung auslöst.

- Für den Fall, dass das aktuelle Tempo oder das Zieltempo < 0 km/h ist, soll in jedem Fall eine Notbremsung ausgelöst werden.
- Für den Fall, dass das aktuelle Tempo größer als 130 km/h ist, soll der Tempomat bremsen.
- Der Tempomat soll bremsen, falls das aktuelle Tempo mehr als 2% größer ist als Zieltempo.
- Der Tempomat soll beschleunigen, falls das aktuelle Tempo mehr als 2% kleiner ist als das Zieltempo.
- Andernfalls (aktuelles Tempo $\pm 2\%$ vom Zieltempo) soll der Tempomat nichts tun.

Führen Sie die im Skript unter Recipe for Intervals beschriebenen Schritte durch. Verwenden Sie die Template-Datei cruise-control.pf. Bearbeiten Sie die mit **todo** markierten Stellen. Geben Sie mindestens 6 sinnvolle Beispiele (Eingaben und erwartete Resultate) in Form von Testfällen an.

## Aufgabe 3: Glückliche Zahlen (1 Punkt)

Schreiben Sie eine Funktion, die Ihnen die glücklichen Zahlen (https://de.wikipedia.org/wiki/Glückliche_Zahl) bis zu einer Obergrenze liefert.

a) Schreiben Sie eine Funktion fill-array, die eine Zahl **n** übergeben bekommt und ein Array mit den Zahlen von 1 bis einschließlich **n** füllt und auf dem Stack ablegt.

b) Schreiben Sie eine Funktion lucky-numbers die eine Zahl n übergeben bekommt und alle Zahlen ausgibt, die glückliche Zahlen im Intervall [0,**n**] sind. Nutzen Sie ihre in a) implementierte Funktion. Testen Sie Ihre Implementierung für die ersten 100 natürlichen Zahlen und vergleichen Sie die Ausgabe mit der Liste auf Wikipedia.

c) Was hat es mit dem Siebprinzip auf sich? Nennen Sie ein Beispiel für andere bekannte Zahlen, die mit einem Siebprinzip gewonnen werden können.

## Aufgabe 4: Auktionshaus (1 Punkt)

Ein Prototyp für ein Auktionshaus soll in Postfix implementiert werden (**auction.pf**). Auktionen werden als **Objekt** gespeichert, welches eine von drei Alternativen darstellt.

Eine Auktion kann vom Typ :Int sein, wenn Sie noch nicht begonnen hat. Dann entspricht der Wert dem Startzeitpunkt der Auktion (in Sekunden).

Weiter kann sie vom Typ :Arr sein, dann enthält das Array das aktuelle Gebot, den aktuellen Bieter und den Endzeitpunkt (in Sekunden).

Wenn die Auktion beendet ist, so ist die Auktion vom Typ :Str und enthält den Namen des Höchstbietenden.

**Beispiele dazu sind im Code genannt.** Vervollständigen Sie die folgenden Funktionen.

a) Implementieren Sie die Funktion **bid**: (auction :Obj, bidder :Str, offer :Num -> :Obj), um das Bieten auf eine Auktion zu ermöglichen.

   Ein Gebot soll in folgenden Fällen nicht möglich sein:

   - Wenn die Auktion noch nicht begonnen hat oder beendet ist
   - Wenn das neue Gebot kleiner oder gleich dem alten Gebot ist
   - Wenn der gleiche Bieter versucht sein Gebot zu erhöhen
   - Wenn der Bieter der leere String ist ""

   In den oben genannten Fällen soll das alte Gebot erhalten bleiben. Ansonsten wird das Gebot aktualisiert (die Endzeit bleibt immer gleich). Schauen Sie sich dazu auch die Testfälle an.

b) Implementieren Sie die Funktion **update-auction**: (auction :Obj, time :Int -> :Obj). Diese soll den Zustand einer Auktion anhand einer übergebenen Zeit aktualisieren.

   Wenn die übergebene Zeit (Parameter time) gleich oder größer als die Startzeit ist, soll die Auktion initialisiert und gestartet werden. Der Endzeitpunkt entspricht der Startzeit plus der Auktionszeit. Ab dem Endzeitpunkt soll die Auktion beendet werden. **Beachten Sie die Konstanten am Anfang des Codes.**

c) **(OPTIONAL)** Wenn alle Testfälle fehlerfrei funktionieren, kommentieren Sie die letzte Zeile aus und starten Sie das Programm:
   ```
   #todo: remove comment to animate
   #animate
   ```

   Bieten Sie ein paar Mal in dem Sie auf den Button "**Bieten!**" klicken und prüfen Sie, ob Ihre Implementierung funktioniert. Die Eingabe des Namens und des Gebots sind leider aktuell nur über die Eingabe der Postfix IDE möglich. Bitte beenden Sie die Applikation immer über den roten Stopp-Button in der IDE.