Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 – WS 2020/21

Prof. Dr. Michael Rohs, Tim Dünte, M.Sc.

Übungsblatt 4

Alle Übungen (bis auf die erste) müssen in Zweiergruppen bearbeitet werden. Beide Gruppenmitglieder müssen die Lösung der Zweiergruppe einzeln abgeben. Die Namen beider Gruppenmitglieder müssen sowohl in der PDF Abgabe als auch als Kommentar in jeglichen Quelltextabgaben genannt werden. Wenn Sie für Übungsblatt 1 noch keinen Gruppenpartner haben, geben Sie alleine ab und nutzen Sie das erste Tutorium dazu, mit Hilfe des Tutors einen Partner zu finden. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 12.11. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/WiSe2020/Programmieren1. Die Abgabe muss aus einer einzelnen zip-Datei bestehen, die den Quellcode, ein pdf bei Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen bitte auf.

Lesen Sie das PostFix-Tutorial (https://postfix.hci.uni-hannover.de/postfix-lang.html) weiter bis zum Abschnitt Data Definitions (einschließlich).

Aufgabe 1: Skript

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

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

b) Lesen Sie Kapitel 8 (Recipe for Variant Data) und beantworten Sie folgende Frage: Welchen Vorteil bietet `datadef` im Zusammenhang mit Variant Data?

c) Lesen Sie Kapitel 9 (Recipe for Self-Referential Data) und beantworten Sie folgende Frage: Was versteht man unter Induktionshypothese im Zusammenhang mit rekursiven Funktionen?

d) Was war Ihnen beim Lesen der Kapitel 7-9 unklar? Wenn nichts unklar war, welcher Aspekt war für Sie am interessantesten?

Aufgabe 2: Text-Statistik

Entwickeln Sie ein Programm zur Erstellung von Text-Statistiken. Eingabe der Funktion ist eine Zeichenkette beliebiger Länge. Ausgabe ist eine statistische Zusammenfassung der Zeichenkette, mit folgenden Komponenten: Anzahl Zeichen (inkl. Leerraum), Anzahl Buchstaben (a-z bzw. A-Z), Anzahl Ziffern (0-9), Anzahl Sätze (Sätzen enden mit „.“, „!“ oder „?“) und Anzahl Zeilen. Zum Leerraum (whitespace) gehören folgende Steuerzeichen: Leerzeichen (`" "`), Tabulator (`"\t"`), Zeilenvorschub (`"\n"`) und Wagenrücklauf (`"\r"`). Im Eingabetext erlaubt sind Zeichen mit einem ASCII-Code zwischen 32 (Leerzeichen) und 126 (Tilde) sowie Leerraum (also insbesondere sind keine Umlaute erlaubt).

Führen Sie die im Skript unter Recipe for Compound Data beschriebenen Schritte durch, sofern sie nicht bereits vorgegeben sind. Verwenden Sie die Template-Datei `text-statistics.pf`. Bearbeiten Sie die mit `todo` markierten Stellen. Beachten Sie die Hinweise.

Aufgabe 3: Rekursive Listen in Postfix

Eine Liste sei definiert als entweder leer (`null`) oder ein Paar (`pair`) aus einem ersten Element (`pair-first`) und einer Rest-Liste (`pair-rest`). Implementieren Sie die nachfolgend genannten Funktionen rekursiv und ohne Verwendung von Array-Funktionen. Verwenden Sie die in list.pf vorgegebene Datendefinition.

a) Implementieren Sie die Funktion `lprepend`, die `x` vorne vor die Liste `l` einfügt und die resultierende Liste zurückgibt. Schreiben Sie einen Dokumentations-String (purpose statement).

b) Implementieren Sie die Funktion `has-string?`, die prüft, ob die Liste `l` mindestens einen String enthält. Schreiben Sie einen Dokumentations-String (purpose statement).

c) Implementieren Sie die Funktion `only-numbers?`, die prüft, ob die Liste `l` ausschließlich Zahlen enthält. Schreiben Sie einen Dokumentations-String (purpose statement).

d) Implementieren Sie die Funktion `keep-strings`, die nur die in der Liste `l` vorkommenden Strings als Liste zurückgibt. Schreiben Sie einen Dokumentations-String (purpose statement).

e) Implementieren Sie die Funktion `range`, die eine aufsteigende Liste von ganzen Zahlen zurückgibt. Für `3 7 range` soll die Funktion beispielsweise eine Liste mit den Zahlen 3, 4, 5, 6, 7 (in dieser Reihenfolge) zurückliefern. Schreiben Sie einen Dokumentations-String (purpose statement).

Aufgabe 4: C-Compiler installieren

Diese Aufgabe dient der Vorbereitung der C-Aufgaben ab dieser Woche. Installieren Sie auf Ihrem Rechner den GCC-Compiler und einen Texteditor (am besten mit Syntaxhervorhebung). Nähere Erläuterungen zur Installation unter Windows, OS X und Linux finden Sie in den Übungsfolien.

a) Testen Sie die Installation indem Sie `gcc -v` auf der Kommandozeile aufrufen. Fügen Sie die Ausgabe die Sie erhalten in ihre Abgabe ein.

b) Testen Sie Ihre Installation außerdem mit folgendem Programm:

```
1. #include <stdio.h>
2. int main(void) {
3.       printf("hello, world\n");
4.       return 0;
5. }
```

Speichern Sie das Programm als Textdatei unter dem Namen `hello.c`

Öffnen Sie eine Kommandozeile und wechseln Sie in das Verzeichnis, in dem `hello.c` liegt.:

`cd /Pfad/zu/meinem/Verzeichnis` bzw.

`cd C:\Pfad\zu\meinem\Verzeichnis`

Kompilieren Sie das Programm:
`gcc hello.c -o hello` bzw. `gcc hello.c -o hello.exe`

Führen sie das Programm aus:
`./hello` bzw. `hello.exe`

Die Ausgabe sollte lauten:
`hello, world`

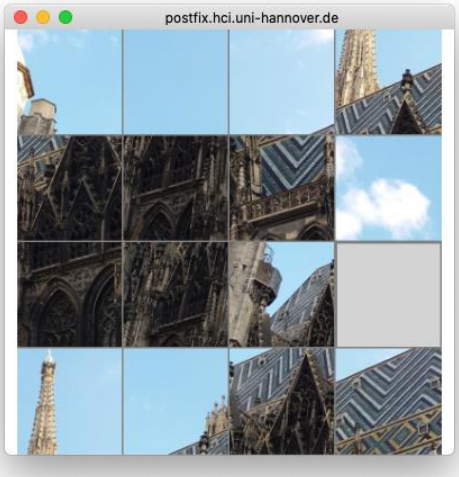
c) Erweitern Sie das Programm, so dass statt „world" ihr Name ausgegeben wird. Bspw.:
`hello, Tim Duente`
Das Programm `hello.c` ist Teil der Abgabe.

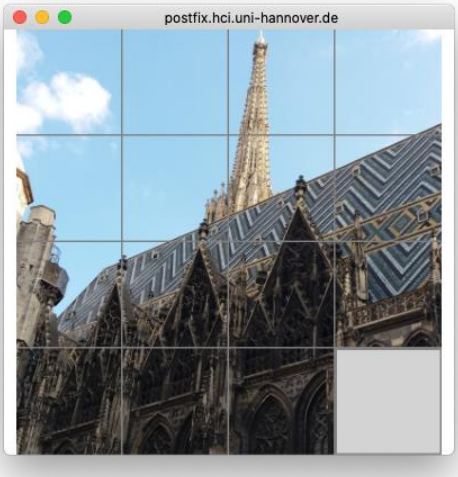
d) Welche Funktion haben die folgenden Terminal Befehle? Geben Sie einen Beispielaufruf an und fügen Sie, falls es eine Ausgabe auf der Konsole gibt, diese der Abgabe bei:
   1. mkdir
   2. ls
   3. cd

e) Starten Sie ein neues Terminal und stellen Sie sicher in welchem Verzeichnis Sie sind. Kopieren Sie nun mit einem Filemanager die Zip-Datei task4.zip in dieses Verzeichnis. Geben Sie nun im Terminal „ls" ein um zu prüfen, ob Sie die Zip-Datei an die richtige Stelle kopiert haben. Entpacken Sie die Datei mit dem unzip Kommando im Terminal. Es werden eine Reihe von Dateien und Ordnern entpackt.
   1. Geben Sie zwei verschiedene Wege an wie Sie mit cd in das Verzeichnis folder/A01 gelangen können.
   2. Geben Sie zwei verschiedene Wege an wie Sie mit cd vom Verzeichnis A01 in das Verzeichnis A02 gelangen.
   3. Was passiert, wenn Sie in Ihrem ursprünglichen Verzeichnis cd folder/../folder/A01 eingeben?
   4. Was passiert, wenn Sie statt ls „ls -al" aufrufen?

Aufgabe 5: 15-Puzzle (1 Bonuspunkt)

Implementieren Sie die fehlenden Funktionen des 15-Puzzles. Eine detaillierte Beschreibung des Spiels findet sich hier: https://de.wikipedia.org/wiki/15-Puzzle. Die Template-Datei ist puzzle.pf. Das Ziel des Spiels ist in folgender Abbildung illustriert:



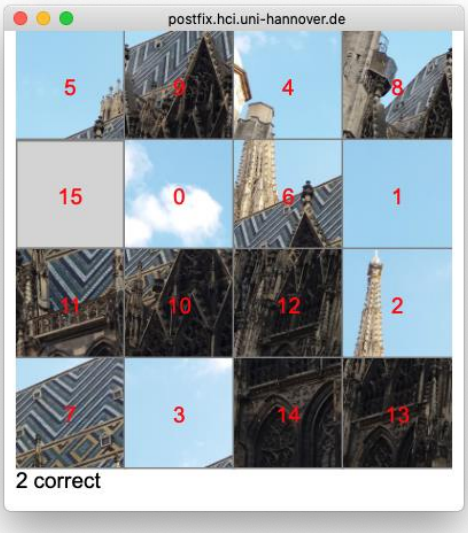


Der Zustand auf der linken Seite soll durch Verschieben von Kacheln in den sortierten Zustand auf der rechten Seite überführt werden. Kacheln können durch Klicken mit der Maus sowie mittels der Pfeiltasten verschoben werden. Durch Klicken können auch mehrere Kacheln in einer Reihe oder Spalte auf einmal verschoben werden. Die Stellen, an denen die einzelnen Teilaufgaben zu implementieren sind, sind in der Template-Datei markiert.

a) Vervollständigen Sie die Funktion `position-to-index: (x :Int, y :Int -> :Int)`, die eine Position in einen Array-Index konvertiert. Links oben ist die Position (0, 0), rechts daneben Position (1,0), usw. Rechts unten ist die Position (3, 3). Die Kacheln sind durch ganzzahlige Werte einem Array der Länge 16 repräsentiert. In der sortierten Reihenfolge entspricht der Wert der Indexposition. Beachten Sie dazu auch die Erläuterung im Kommentar in puzzle.pf.

b) Implementieren Sie die Funktion `swap-tiles: (tiles :Arr, x1 :Int, y1 :Int, x2 :Int, y2 :Int -> :Arr)`. Diese tauscht die Kachel an der Position (x1, y1) mit der Kachel an der Position (x2, y2).

c) Implementieren Sie die Funktion `count-correct: (tiles :Arr -> :Int)`. Diese gibt die Anzahl der korrekt positionierten Kacheln zurück. Das Puzzle ist gelöst, wenn die implementierte Funktion 16 zurückgibt, wenn also alle Kacheln auf dem richtigen Platz sind. Hinweis: Es ist zum Verständnis hilfreich, die sortierte Situation zu betrachten. Dies ist durch Auskommentieren von `shuffle` möglich.



d) Implementieren Sie die Funktion `key-press: (state :Arr, key :Str -> :Arr)` so, dass das Spiel nicht nur mit der Maus, sondern auch mit den Pfeiltasten gespielt werden kann. Hinweis: Die Pfeiltasten sind über die Strings „ArrowRight“, „ArrowLeft“, „ArrowUp“ und „ArrowDown“ codiert.

e) Im Template wird die aktuelle Anzahl korrekter Positionen im Output-Bereich ausgegeben.

Implementieren Sie eine grafische Ausgabe der Anzahl korrekter Positionen (wie links abgebildet).

f) Das Spiel ist leichter, wenn die Werte der Kacheln eingeblendet werden (wie links abgebildet). Implementieren Sie die Möglichkeit mit Druck auf die „w"-Taste, die Werte ein- und auszublenden.