Human-Computer Interaction Group

Leibniz Universität Hannover

Übungsblatt 4

Programmieren 1 – WiSe 22/23

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

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. Plagiate führen zum Ausschluss von der Veranstaltung.

Abgabe bis Donnerstag den 10.11. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/WiSe2022/Prog1. Die Abgabe muss aus einer einzelnen Zip-Datei bestehen, die den Quellcode, ein PDF für Freitextaufgaben und alle weiteren nötigen Dateien (z.B. Eingabedaten oder Makefiles) enthält. Lösen Sie Umlaute in Dateinamen auf.

Aufgabe 1: Volumen geometrischer Körper

In einem Programm sollen verschiedene Formen von geometrischen Körpern, nämlich Zylinder, Kugel und Quader, repräsentiert werden. Entwickeln Sie eine Funktion, die diese geometrischen Körper verarbeiten kann (Parametertyp: Shape) und das zugehörige Volumen berechnet.

Führen Sie die im Skript unter Recipe for Variant Data beschriebenen Schritte durch. Verwenden Sie die Template-Datei **volume.pf**. Bearbeiten Sie die mit **todo** markierten Stellen. Geben Sie je ein Beispiel (Eingaben und erwartete Resultate) für jede Variante an.

Aufgabe 2: Rekursive Listen in PostFix

Eine rekursive 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. Nutzen Sie die in **lists.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) **(OPTIONAL)** Implementieren Sie die Funktion **rev-rec** (für reverse recursively), die von der existierenden Funktion rev (für reverse) aufgerufen wird, um eine Liste umzudrehen. Die Variable result dient der Speicherung und Übergabe der bereits umgedrehten Teilliste.

Aufgabe 3: CO_2-Ausstoß von verschiedenen Fahrzeugen

In einem Programm sollen für verschiedene Fahrzeuge, nämlich Fahrräder, E-Roller und Autos, der CO2 Ausstoß in Abhängigkeit von verschiedenen Parametern berechnet werden. Entwickeln Sie eine Funktion, die diese verschiedenen Fahrzeuge verarbeiten kann (Parametertyp: Vehicle) und abhängig von der Laufleistung in km den zugehörigen CO_2-Ausstoß berechnet.

Fahrzeuge haben folgende Parameter:

- Fahrrad: (keine)
- E-Roller: Batteriekapazität (Bsp. 12000mAh)
- Autos: Gewicht, Anzahl Zylinder

Für die Fahrzeuge gelte folgender CO2 Ausstoß je Kilometer:

- Fahrrad: 0
- E-Roller: 0.1 * Kilometer + 20 * Anzahl Ladevorgänge (500mAh Verbrauch je Kilometer)
- Auto: Gewicht * Gewicht * 0.0001 * Anzahl-Zylinder * Kilometer

Führen Sie die im Skript unter Recipe for Variant Data beschriebenen Schritte durch. Verwenden Sie die Template-Datei **co2.pf**. Schauen Sie sich vor Beginn der Bearbeitung den bestehen Code an. Dieser enthält Hinweise, wie Sie die Varianten benennen sollten. Bearbeiten Sie die mit todo markierten Stellen.

Aufgabe 4: C-Compiler installieren

Diese Aufgabe dient der Vorbereitung der C-Aufgaben ab nächster 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
```

Führen Sie das Programm aus:

```
./hello bzw. hello.exe
```

Die Ausgabe sollte lauten:
hello, world

Fügen Sie einen Screenshot der Ausführung Ihrer Abgabe bei.

c) 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

d) 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 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 dem **cd** vom Verzeichnis **A01** in das Verzeichnis **A02** gelangen können.
3. Was passiert, wenn Sie in Ihrem ursprünglichem Verzeichnis **cd folder/../folder/A01** eingeben?
4. Was passiert, wenn Sie statt ls „ls -al" aufrufen?