Übungs-Online-Test C/Python

Ihr Test-Username: u177 **Password:** hknh

Hinweis 1. Erlaubte Hilfsmittel sind die im Teilnehmeraccount bereitgestellten PDF-Dateien (mit Menüpunkt "Skript" bzw. "Python Library" anzeigbar) und das Online-Manual (mit `man`). Nicht erlaubt sind insbesondere Handy-/Internet-Nutzung oder eigene elektronische oder Papier-Unterlagen.

Hinweis 2. Mit `pruef a<Nummer>.c` bzw. `pruef a<Nummer>.py` (also z.B. `pruef a1.c`) können Sie Ihre Lösung automatisiert und unverbindlich vor-testen lassen. Beachten Sie bitte die ausgegebenen Meldungen. Mit `pruef log` können Sie sich das letzte Prüfprotokoll noch einmal ausgeben lassen. Das Prüfprogramm legt Backups halbwegs erfolgreicher getesteter Programme im Verzeichnis `backup.pruef` ab. Sie können während des Tests darauf zurückgreifen, gewertet werden aber nur die `a1.c` bzw. `a2.py` in den jeweils vorgegebenen Verzeichnissen.

Lösungen müssen fehler- und warnungsfrei compilieren **und** ohne Laufzeit- und Speicherfehler die erwarteten Ergebnisse in der vorgegebenen Form liefern. Die Lösungen müssen gemäß Aufgabenstellung allgemein berechnet werden – die Erzeugung von Ergebnissen anhand fixer Test-Ein-/Ausgabewerte (z.B. entlang der Prüfprogramm-Ausgabe) gilt als Täuschungsversuch, die ganze Aufgabe wird dann nicht gewertet.

Hinweis 3. Ihnen stehen in der Toolbar (untere Bildschirmkante) **zwei Terminal**-Symbole zur Verfügung. Das linke (oder Menü: "TerminalHome") öffnet eine Shell für das **C-Arbeitsverzeichnis** (= Login-Verzeichnis), die rechte (Menü "TerminalEclipse") eine für das **Python-Arbeitsverzeichnis** (= Projekt "eclipse" im Eclipse-Workspace `workspace`). Bitte legen Sie alle C-Dateien im Homedirectory und alle Python-Dateien auf der obersten Ebene im genannten Eclipse-Projekt ab. Legen Sie bitte **keine** anderen Projekte, Unterverzeichnisse, Packages etc. in Eclipse an. Zum Starten übersetzter C-Programme dürfen Sie hier **kein "./"** vor den Dateinamen setzen – der Dateiname der übersetzten Datei genügt zum Starten (also nicht "./a.out", sondern nur "a.out").

Aufgabe 1. Bitte lösen Sie diese Aufgabe mit ANSI-C90 in einer Datei `a1.c` in Ihrem Login-Verzeichnis. Implementieren Sie den in der Datei `a1.h` vorgegebenen Funktionsprototyp. Zu Testzwecken können Sie zudem eine eigene `main()`- und ggf. weitere Hilfsfunktionen definieren und auch debuggen. Bitte verändern oder löschen Sie vorgegebenen Header-Dateien **nicht**. Die zu implementierenden Funktionen dürfen keine eigene (Standard-)Ausgabe erzeugen (eine eventuelle eigene `main()`-Funktion darf). Zum Starten Ihres C-Programms geben Sie den Programmnamen direkt ein (ohne führendes "`./`")

Eine einfach verkettete Liste besteht aus Knoten folgender Struktur:

```
1 typedef struct lst { char name[20]; int alter ; struct lst *next; } ListEle ;
```

Jeder Knoten enthält neben dem C-String `name` auch ein ganzzahliges `alter`. Der `NULL`-Zeiger markiert das Listenende bzw. repräsentiert die leere Liste. Jeder einzelne Listenknoten wird dynamisch im Freispeicher allokiert. Ihre Lösung muss ohne `valgrind`-Beanstandungen durchlaufen.

Bitte implementieren Sie folgende Funktionen für diesen Listentyp:

```
1 ListEle *einfuegen(ListEle *lst , const char *n, int a);
2 int dopple(ListEle *lst , const char *n, int a);
3 void befreie (ListEle *lst );
```

Die Funktion `einfuegen()` fügt einen neuen Knoten mit dem übergebenem C-String Namen `n` und Alter `a` am Ende der übergebenen Liste `lst` ein und liefert das erste Element der Liste zurück.

Die Funktion `dopple()` sucht das erste Listenelement mit Namen `n` und Alter `a` und fügt direkt hinter diesem eine Kopie dieses Listenelements ein, wie im Beispiel gezeigt. Rückgabewert ist 1, wenn das gesuchte Element gefunden wurde, ansonsten 0.

Die Funktion `befreie()` gibt den für die übergebene Liste allokierten Speicher wieder frei.

Beispiel: Eingefügt werden (in dieser Reihenfolge) diese Namen,Alter:

```
Joghurta,21 Wubbo,75 Glogomir,12
```

Die resultierende Liste (mit Name und Alter) sieht dann so aus:

```
[Joghurta,21]->[Wubbo,75]->[Glogomir,12]->NULL
```

Auf diese Liste `lst` angewandt würde `dopple(lst,"Wubbo",75)` den Wert 1 liefern, die Liste sieht danach so aus:

```
[Joghurta,21]->[Wubbo,75]->[Wubbo,75]->[Glogomir,12]->NULL
```

Ein nachfolgender Aufruf `dopple(lst, "Joendhard",17)` würde den Wert 0 liefern und die Liste bliebe unverändert, weil kein passender Knoten in `lst` ist.

Aufgabe 2. Bitte lösen Sie diese Aufgabe mit Python 3. Die Lösung ist in die vorgegebene Datei `a2.py` im vorgegebenen Eclipse-Projekt einzutragen. Sie sollten in Eclipse **zu Beginn** einen **Refresh** auf dem vorgegebenen Projekt "eclipse" durchführen, um sicher zu stellen, damit alle Dateien korrekt angezeigt werden. Zu Testzwecken können Sie die vorgegebene `main()`-Funktion ausfüllen und ggf. weitere Hilfsfunktionen ergänzen. Bitte achten Sie darauf, dass beim **Import** Ihrer Module nur die benötigten Klassen/Funktionen bereitgestellt (definiert) werden dürfen und **kein weiterer eigener Code ausgeführt** werden darf, insbesondere dürfen außer der Lösung im unten gezeigten Format **keine Ausgaben** gemacht werden (weder beim Import noch im Funktionsablauf).

In Ihrem Startverzeichnis finden Sie vier Dateien `bestellungen-....txt`, welche Zeilen mit Bestellungen enthält (semikolongetrennt Postleitzahl (PLZ), Kundenname und ganzzahliger Einkaufswert). Beispiel:

```
65197;Zorkel;10
20113;Meier;42
65197;Meier;17
65197;Huber;85
47368;Glosema;5
20113;Arlykkendorf;16
65197;Meier;50
```

Bitte schreiben Sie in `a2.py` eine Python-Funktion `statistik(dateiname)`, die welche die Datei mit dem übergebenen Namen `dateiname` liest, verarbeitet und auf der Standardausgabe je eine Zeile je PLZ ausgibt mit den dortigen Kunden und (in Klammern) der jeweiligen Einkaufs**summe**. Wenn je Postleitzahl ein Name mehrfach auftritt, handelt es sich um dieselbe Person. Die Zeilen sollen dabei nach PLZ **aufsteigend sortiert** und die Kunden je Zeile nach Namen **aufsteigend sortiert** ausgegeben werden. Das Ergebnis muss wie in folgendem Beispiel formatiert sein (und natürlich soll das Programm auch mit anderen Bestelldateien gleichen Aufbaus analog funktionieren). Die Funktion darf darüber hinaus keine weiteren Ausgaben erzeugen.

```
20113: Arlykkendorf(16) Meier(42)
47368: Glosema(5)
65197: Huber(85) Meier(67) Zorkel(10)
```