It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.
Sir Arthur Conan Doyle, Sherlock Holmes

UNIVERSITÄT DES SAARLANDES

FR Mathematik
Andreas Buchheit

7. Übung zur Vorlesung
Programmierung
im Sommersemester 2019

Abgabe: Mittwoch, den 05.06.2019 bis spätestens 12 Uhr.

Aufgabe 7.1. (12 Punkte) Data Server
Im Folgenden sollen Sie die Allozierung, die Verwendung und das Freigeben von dynamischen Arrays lernen in dem Sie eine Datenbank für Lognachrichten schreiben.
Sie sollen nun einen rudimentären Speicher für Lognachrichten implementieren, welcher automatisch mit der Anzahl an gespeicherten Nachrichten anwächst.

(a) Schreiben Sie hierzu eine Funktion

```
char** new_database();
```

die ein Array von `N_BASE == 5` Pointern auf dem Heap alloziert und eine Prozedur

```
void free_database(char** database);
```

die ein Array von `N_BASE` Pointern freigibt.

(b) Schreiben Sie dann eine Routine

```
char** add_message(char* message, char** database, size_t* ptr_n);
```

welche einen nullterminierten String `message` der Datenbank hinzufügt. Hierzu wird die Länge des Strings (ohne das terminierende Nullbyte) mittels `strlen` aus `string.h` besimmt. Daraufhin wird ein Array alloziert, das genau der Länge des Strings (mit Nullbyte) entspricht und der Pointer zu dem Speicherplatz wird in dem Pointerarray festgehalten. Der Pointer der Datenbank wird zurückgegeben (er kann sich beim reallozieren ändern) und die Anzahl an Nachrichten, die an der Speicheradresse `ptr_n` gespeichert ist, wird inkrementiert.

Die momentane Anzahl an Nachrichten wird an der Speicheradresse `ptr_n` gesichert. Entspricht die Anzahl an momentanen Nachrichten `*ptr_n` einem Vielfachen von `N_BASE`, so soll `add_message` das Pointerarray mithilfe von `realloc` um `N_BASE` Einträge vergrößern.

Schreiben Sie daraufhin eine entsprechende Routine

```
char** delete_last_message(char** database, size_t* ptr_n);
```

die den Speicherplatz der letzten Nachricht freigibt und die Anzahl an Nachrichten um eins erniedrigt. Ist `*ptr_n` nach Löschen der letzten Nachricht ein Vielfaches von `N_BASE`, so verkleinere das Pointerarray um `N_BASE` Elemente.

(c) Implementieren Sie eine Prozedur

```
void print_message(char** database, size_t i, size_t n);
```

die die `i`-te Nachricht ausgibt, oder eine Fehlermeldung, falls `i>n`. Implementieren Sie daraufhin eine neue Funktion

```
char** read_and_add_line(char** database, size_t* ptr_n);
```

die eine Zeile aus dem Userinput einliest, in einem Bufferarray der Größe `BUFSIZ` zwischenspeichert und daraufhin als neue Nachricht in der Datenbank sichert (und erneut den Pointer auf die Datenbank zurückgibt).

(d) Schreiben sie eine Prozedur, die das komplette Log ausgibt. Schreiben Sie dann eine weitere Prozedur, die alle Lognachrichten und daraufhin die Datenbank selbst freigibt. Testen Sie ihre Datenbank!