Programmentwurf

In dem Programmentwurf soll ein Terminplaner als C-Programm implementiert werden. Das Programm verwaltet eine Liste mit Terminen und bietet die folgenden Funktionen an:

- Das Anzeigen aller Termine am aktuellen Tag.
- Das Anzeigen aller Termine an einem bestimmten Tag, den der/die BenutzerIn von der Konsole eingibt.
- Das Anzeigen der gesamten Liste.
- Das Einfügen eines neuen Termins.
- Das Suchen nach einem Termin in der Liste.
- Das Löschen eines Termins in der Liste.
- Das Löschen der gesamten Liste.
- Das Beenden des Programms.

Die Teminliste soll vor dem Beenden des Programms in einer Datei gespeichert werden. Beim Starten des Programms muss die Datei wieder eingelesen und automatisch alle bereits abgelaufenen Termine gelöscht werden. Anschließend werden alle Termine am aktuellen Tag ausgegeben.

Implementierung

Für die Implementierung soll eine C-Struktur `Appointment` der Form

```
typedef struct
{
  time_t  start;
  char   *description;
} Appointment;
```

zum Speichern eines Termins verwendet werden. Die Termine werden in einer verketteten Liste aufsteigend sortiert nach dem Zeitpunkt gespeichert. Zur Speicherung der verketteten Liste sollen zwei C-Strukturen `Element` und `List` der Form

```
typedef struct Element
{
  Appointment *appointment;
  struct Element *next;
} Element;

typedef struct
{
  Element *head, *tail;
} List;
```

eingesetzt werden. Die erste Struktur speichert ein Listenelement zusammen mit einem Zeiger auf den Nachfolger, die zweite Struktur speichert den Anfang und das Ende der Liste.

Weiterhin sind die folgenden C-Funktionen zu implementieren:

(a) Sieben Funktionen zum Verwalten und Ausgeben der verketteten Liste.

- `createList` hat keine Parameter und erstellt eine leere Liste, die als Struktur `List` zurückgegeben wird.
- `clearList` hat einen Parameter vom Typ `List` und gibt den allokierten Speicher der Liste frei. Die beiden Pseudo-Elemente dürfen nicht gelöscht werden.
- `insertElement` erhält eine Liste (als Struktur `List`), einen Zeitpunkt (als Datentyp `time_t`) und einen Text (als Datentyp `const char *`) als Parameter und speichert die Daten in einer Struktur vom Typ `Appointment`, die dann sortiert nach dem Zeitpunkt in die Liste eingefügt wird. Für den Text soll die Funktion ebenfalls Speicher allokieren und den String im Übergabeparameter in den Speicherbereich kopieren.
- `findElement` erhält eine Liste (als Struktur `List`) und einen Text (als Datentyp `const char *`) und sucht nach dem Text in der Liste. Ist der Eintrag in der Liste vorhanden, dann wird ein Pointer auf das Element zurückgegeben. Andernfalls wird ein NULL-Pointer zurückgegeben.
- `deleteElement` erhält eine Liste (als Struktur `List`) und einen Text (als Datentyp `const char *`) und sucht nach dem Text in der Liste. Ist der Eintrag in der Liste vorhanden, dann wird er gelöscht und der Wert `true` zurückgegeben. Andernfalls wird nichts geändert und der Wert `false` zurückgegeben.
- `printAppointment` erhält einen Pointer auf eine Struktur vom Typ `Appointment` und gibt die Daten in der Struktur in einer Zeile auf der Konsole aus.
- `printList` erhält eine Liste (als Struktur `List`) und drei Parameter vom Typ `int`, in denen der Tag, Monat und das Jahr eines Datums übergeben werden. Die Funktion soll alle Termine an diesem Tag auf der Konsole ausgeben. Haben alle drei Parameter den Wert 0, dann soll die gesamte Liste ausgegeben werden.

(b) Die Funktion `menu` zeigt die acht Menüpunkte an und liest eine Eingabe von der Konsole ein. Ist die Eingabe gültig, dann werden anschließend weitere Parameter (sofern notwendig) eingelesen und die zugehörige Operation durch den Aufruf einer geeigneten Funktion aus Teilaufgabe (a) ausführt. Bei einer falschen Eingabe soll eine Fehlermeldung erfolgen. Die Funktion wird erst beendet, wenn die Menüoption zum Beenden des Programms ausgewählt wurde.

(c) Die `main`-Funktion liest die Datei ein und speichert diese in einer verketteten Liste, abgelaufene Termine werden dabei ausgelassen. Der Name der Datei wird als Kommandozeilenparameter übergeben. Fehlt der Parameter, dann soll der Dateiname "termine.txt" verwendet werden. Falls die Datei nicht eingelesen werden kann, wird eine leere Liste erstellt. Nach dem Einlesen der Liste werden die Termine für den aktuellen Tag anzeigt und das Startmenü durch den Aufruf der Funktion `menu` anzeigt. Vor dem Beenden des Programms wird die Liste in der Datei gespeichert und anschließend die gesamte Liste einschließlich der beiden Pseudo-Elemente freigegeben.

Bedingungen

- Eine Bearbeitung in Gruppen zu 3 Studierenden ist möglich. Alle Studierenden einer Gruppe erhalten die gleiche Bewertung.
- Das Programm muss unter Linux mit dem GCC-Compiler unter Verwendung der Optionen `-Wall` und `-pedantic-errors` fehlerfrei und ohne Warnungen kompilierbar sein.
- Die vorgegebenen Strukturen und Funktionen müssen wie oben beschrieben zur Lösung der Aufgabe eingesetzt werden. Es dürfen nach Bedarf weitere Funktionen ergänzt werden, aber keine globalen Variablen.
- Es dürfen alle Funktionen der C-Standard-Bibliothek verwendet werden. Zur Konsolenausgabe sind nur die Zeichen der Standard-ASCII-Tabelle (Werte 0-127) erlaubt.
- Abgabe: Source-Code als c-File oder zip-File auf Moodle hochladen. Das File mit den Namen der Gruppenmitglieder benennen. Abgabetermin ist 05.03.2023.

Bewertungskriterien

Teilaufgabe (a)	16 Punkte
Teilaufgabe (b)	12 Punkte
Teilaufgabe (c)	12 Punkte
Kompilierbarkeit und Robustheit	5 Punkte
Strukturierung und Kommentierung	5 Punkte
Gesamt	50 Punkte