TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professur
Betriebssysteme

Algorithmen und Programmierung

Typ String (Trainingsaufgabe)

In dieser Aufgabe sollen verschiedene Funktionen zur Bearbeitung von Zeichenketten implementiert werden. Sie dürfen dabei keine Funktionen der Bibliothek `string.h` verwenden. Nutzen Sie `malloc` zum Allozieren von neuem Speicher in den Funktionen, in denen dies benötigt wird. Um die Freigabe des Speichers müssen Sie sich nicht kümmern, dies übernimmt der Rufende der Funktion. Die Semantik der entsprechenden Funktionen sei wie folgt gegeben:

1. `int stringLength (char *string)` liefert die Länge einer gegebenen Zeichenkette, wenn diese nicht `NULL` ist. Sonst wird der Wert `-1` zurückgegeben.
2. `bool equals (char *a, char *b)` liefert `true`, wenn die Zeichenketten `a` und `b` gleich lang sind und die Zeichen an den gleichen Positionen übereinstimmen, sonst `false`.

   Beispiele:

   - `equals("ab", "ab")` ⇒ `true`
   - `equals("aB", "ab")` ⇒ `false`
   - `equals("ab", "abc")` ⇒ `false`
   - `equals(NULL, NULL)` ⇒ `false`
   - `equals("", "")` ⇒ `false`
3. `char* substring(char *string, unsigned int from, unsigned int to)` liefert den Teil der Zeichenkette `string`, der bei Index `from` beginnt und bei Index `to` endet. Bei ungültigen Parametern soll `NULL` zurückgegeben werden. Beachten Sie, dass `string` durch die Funktion nicht verändert werden darf.

   Beispiele:

   - `substring("Hello", 3, 10)` ⇒ `"lo"`
   - `substring("Hello", 6, 0)` ⇒ `NULL`
   - `substring("Hello", 6, 7)` ⇒ `NULL`
4. `int findSubstr(char *string, char *substring)` findet das erste Vorkommen einer Zeichenkette `substring` in der Zeichenkette `string` und liefert die Position des ersten Elementes von `substring` in `string` zurück. Im Fehlerfall soll `-1` zurückgegeben werden.

Beispiele:

- `findSubstr("Hello Wally", "Wally") -> 6`
- `findSubstr("Hello Wally", "foo") -> -1`

5. `char* concat(char *s1, char *s2)` liefert eine neue zusammengesetzte Zeichenkette aus `s1` und `s2`. Wenn eine der Eingaben NULL ist, muss auch das Ergebnis NULL sein.

   Beispiele:

   - `concat("abc", "def")` ⇒ `"abcdef"`
   - `concat("abc", "")` ⇒ `"abc"`
   - `concat("", "abc")` ⇒ `"abc"`
   - `concat("abc", NULL)` ⇒ `NULL`
   - `concat(NULL, NULL)` ⇒ `NULL`

6. `char** splitString(char *string, char splitter)` zerteilt eine Zeichenkette `string` in einen Array von neuen Teilzeichenketten. Letztere werden durch das übergebene Trennzeichen `splitter` identifiziert.

   Beispiele:

   - `splitString("hallo welt", ' ')` ⇒ `["hallo", "welt"]`
   - `splitString("foo;bar", ';')` ⇒ `["foo", "bar"]`
   - `splitString("foo;bar", ',')` ⇒ `["foo;bar"]`
   - `splitString("", '-')` ⇒ `[""]`
   - `splitString(NULL, '-')` ⇒ `[NULL]`

7. `char* replaceAll(char *string, char *substring, char *with)` ersetzt **alle** Vorkommnisse der Zeichenkette `substring` in der Zeichenkette `string` mit einer Zeichenkette `with`. Dabei kann `substring` länger, kürzer oder genauso lang wie `with` sein. Das Ergebnis ist eine neue Zeichenkette, `string` bleibt also unverändert. Im Fehlerfall wird NULL zurückgegeben.

   Beispiele:

   - `replaceAll("Hallo", "l", "abc")` ⇒ `"Haabcabco"`
   - `replaceAll("Hallo", "ll", "")` ⇒ `"Hao"`
   - `replaceAll("aaa", "aa", "bb")` ⇒ `"bba"`
   - `replaceAll("Hallo", 'll', NULL)` ⇒ `NULL`

8. `char* toLowerCase(char *string)` liefert eine neue Zeichenkette zurück, in der alle groß geschriebenen Buchstaben in `string` durch den entsprechenden Kleinbuchstaben ersetzt werden.

   Beispiele:

- `toLowerCase("AbL1ca")` ⇒ `"abl1ca"`
- `toLowerCase(NULL)` ⇒ `NULL`

9. `char* toUpperCase(char *string)` liefert eine neue Zeichenkette zurück, in der alle klein geschriebenen Buchstaben in `string` durch den entsprechenden Großbuchstaben ersetzt werden.

   Beispiele:

   - `toLowerCase("AbL1ca")` ⇒ `"ABL1CA"`
   - `toLowerCase(NULL)` ⇒ `NULL`

10. `int numOfOccurrences(char *string, char c)` liefert die Anzahl der Vorkommnisse des Zeichens `c` in der Zeichenkette `string`. Wenn `string NULL` ist, so soll die Rückgabe `-1` sein.

    Beispiele:

    - `numOfOccurrences("Hallo Welt!", 'l')` ⇒ `3`
    - `numOfOccurrences(NULL, 'a')` ⇒ `-1`

Alle genannten Funktionen sollen in der Datei `stringType.c` implementiert werden. Die `stringType.h` steht für Sie bereit und ist wie folgt definiert:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int stringLength (char *);
bool equals (char *, char *);
char* substring(char *, int, int);
int findSubstr(char *, char *);
char* concat(char *, char *);
char** splitString(char *, char);
char* replaceAll(char *, char *, char *);
char* toUpperCase(char *);
char* toLowerCase(char *);
int numOfOccurrences(char *, char);
```

Listing 1: stringType.h

Reichen Sie Ihre Lösung als Datei `stringType.c` ein. Es wird kein `Makefile` und keine Implementierung von `main()` benötigt.

Hinweise zur Aufgabenstellung

Für die Lösung dieser Aufgabe benötigen Sie folgende Grundkenntnisse. Im in Klammern angegebenen Foliensatz finden Sie Informationen dazu.

- Kontrollfluss (`if`) in C (Kapitel III)

- Funktionen in C (Kapitel III)
- Schleifen in C (Kapitel V)
- Felder in C (Kapitel IV)
- Dynamische Speicherverwaltung (`malloc()`) (Kapitel IV)
- Benutzung von `gcc` (Kapitel III)

Hinweise zur Abgabe

- Erstellen Sie eine ZIP- bzw. TGZ-Archivdatei, welche die geforderten Dateien enthält.
- Fügen Sie dem Archiv keine weiteren Dateien oder Ordner hinzu.
- Reichen Sie Ihre Lösung unter `https://osg.informatik.tu-chemnitz.de/submit` ein.
- Bis zum Abgabeende (Deadline), sofern gegeben, können beliebig neue Lösungen eingereicht werden, die die jeweils älteren Versionen ersetzen.
- Ihr Programm muss auf der Testmaschine übersetzbar sein. Deren Details sind auf dem OpenSubmit-Dashboard verfügbar.
- Ihre Lösung wird automatisch validiert. Sie werden über den Abschluss der Validierung per eMail informiert.