Friedrich-Alexander-Universität Erlangen-Nürnberg

Schriftliche Prüfung zur Vorlesung
“Einführung in die Programmierung für Ingenieure (C)”

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Gesamt
32	8	10	5	5	60

(Name) (Vorname) (Matrikel-Nr.)

(Fachrichtung) (Semester)

Erlangen, 05.10.2000

Organisatorische Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung am Blattende unterschreiben.

- Die Bearbeitungszeit beträgt 60 Minuten.
- Es sind **keine** Hilfsmittel zugelassen.
- Schmierpapier darf **nicht** mit abgegeben werden. Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen freien Raum geschrieben werden, der jeweils der Aufgabe folgt. Sollte der Platz nicht ausreichen, so können Sie die Rückseiten der Aufgabenblätter mitverwenden.
- Bei Bedarf sind von der Aufsicht Schmierpapier (farbig) und zusätzliche Lösungsblätter (weiß) zu erhalten. Vermerken Sie vor dessen Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf! **Schmierpapier darf nicht mit abgegeben werden!**
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studienfachrichtung und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Bitte legen Sie Ihren Studenten- und einen Lichtbildausweis zur Kontrolle bereit.
- Sie dürfen den Raum nicht verlassen, bevor Ihre Personalien überprüft wurden und Sie die Klausurunterlagen der Aufsicht zurückgegeben haben.
- Bleiben Sie bitte am Ende der Bearbeitungszeit solange an Ihrem Platz sitzen, bis alle Klausurunterlagen eingesammelt sind.

Die Klausurergebnisse werden bis Ende der ersten Augustwoche am schwarzen Brett vor dem Sekretariat des Lehrstuhls für Betriebssysteme (vor Raum 0.047 RRZE Erdgeschoss) und im WWW unter folgender URL veröffentlicht: http://www4.informatik.uni-erlangen.de/result00b.html

Erklärung

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen (11 Seiten inkl. Deckblatt).
- die Kenntnisnahme der obigen Informationen.

Ich bin damit einverstanden, daß mein Prüfungsergebnis unter Angabe der Matrikel-Nr. veröffentlicht wird.

☐ ja ☐ nein

Der Schein soll direkt an das Prüfungsamt geschickt werden:

☐ ja ☐ nein

Erlangen, 05.10.2000

Aufgabe 1: (32 Punkte)

Schreiben Sie ein Programm **buecher** zum Erfassen einer beliebigen Anzahl von Büchern.

Für jedes Buch sollen folgende Daten erfasst werden

- Titel (max. 80 Buchstaben lang)
- Buchnummer (als Zahl)
- Verweis auf den Autor

wobei Personendaten so aussehen sollen:

- Vorname (max. 24 Buchstaben lang)
- Nachname (max. 24 Buchstaben lang)

Das Programm wird folgendermassen aufgerufen

buecher ***Ausgabedatei***

und arbeitet dann wie folgt beschrieben:

- Aufrufparameter wird ausgewertet, bei falschem Aufruf Fehlermeldung
- mit Hilfe der Funktion **`neues_buch`** wird eine Datenstruktur zur Speicherung der Buchdaten erzeugt, diese Struktur wird mit Hilfe der Funktion **`erfasse_daten`** gefüllt. Anschließend wird diese Struktur vorne in eine verkettete Liste solcher Strukturen eingehängt (das wird nicht durch `erfasse_daten`, sondern in der Funktion `main` erledigt!).
  Dies wird wiederholt, bis **`erfasse_daten`** das Ende der Eingabe meldet.
- schliesslich werden alle in der Liste gespeicherten Buchdaten mit Hilfe der Funktion **`speichere_buchliste`** in die angegebene Ausgabedatei gespeichert.

Auf den folgenden Seiten finden Sie ein Gerüst für das beschriebene Programm. In den Kommentaren sind die Aufgaben der einzelnen, zu ergänzenden Programmteile beschrieben.


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

/*
 * Deklaration der Personenstruktur
 */
struct person {

     _____________________________________________________________
     _____________________________________________________________

};

/*
 * Deklaration der Buchstruktur
 */
struct buch {

     _____________________________________________________________
     _____________________________________________________________
     _____________________________________________________________

     struct buch *naechstes;
};

/*
 * Deklaration von Funktionen, die bei der Programmierung von
 * main benötigt werden und die im Rahmen dieser
 * Aufgabe (zusätzlich zu main) zu programmieren sind (Seite 6)
 */
struct buch *neues_buch();
void speichere_buchliste(struct buch *, char *);

/*
 * Deklaration von Funktionen, die bei der Programmierung von
 * main benötigt werden, die aber nicht selbst programmiert
 * werden müssen
 */
/* erfasse_daten: liefert 0 am Ende der Eingabe, sonst 1 */
int erfasse_daten(struct buch *);
```

```
/*
 * Funktion main
 */
int main(int argc, char *argv[]) {
       /* Variablendefinitionen */

       /* Überprüfen der Kommandozeile */

       /*
        * Erzeugen von Buchstruktur, Daten erfassen,
        * vorne in Liste einketten
        */

       /*
        * Liste in Ausgabedatei speichern
        */

       return 0;
}
```



```
/*
 * Funktion neues_buch zum Erzeugen einer neuen Buchstruktur
 * (wenn beim Reservieren des Speichers ein Fehler auftritt,
 * gibt diese Funktion eine Fehlermeldung aus und beendet das
 * Programm)
 * Die neue Struktur soll nicht initialisiert werden!
 */
struct buch *neues_buch() {

}
```

```
/*
 * Funktion speichere_buchliste
 * öffnet zuerst die übergebene Datei
 * - wenn hierbei ein Fehler auftritt, wird eine entsprechende
 *   Meldung ausgegeben und das Programm abgebrochen
 * anschliessend wird für jedes Listenelement eine Zeile in der Form
 * Buchnummer, Titel, Autorenname (Vorname und Nachname)
 * in die Datei geschrieben
 */
void speichere_buchliste(struct buch *liste, char *datei) {
      FILE *fp;

      _____________________________________________________________
      _____________________________________________________________

      /* Ausgabedatei öffnen */

      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________

      /* Liste ausgeben */

      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
      _____________________________________________________________
}
```


Aufgabe 2: (8 Punkte)

Markieren und verbessern Sie die Fehler in dem untenstehenden Programm, das eine vom Benutzer einzugebende Zeichenkette rückwärts wieder ausgibt. (Sie können davon ausgehen, daß max. 8 Fehler im Programm versteckt sind.)

Hinweise:

- alle bis auf einen Fehler werden nicht von Standard-Compilern erkannt
- `strlen` ergibt die Länge ohne abschließendes `'\0'`
- `fgets` liest Zeichen in den angegebenen Speicherbereich
- eine fehlende Abfrage auf `NULL` bei `malloc` soll hier nicht als Fehler zählen

Falsch erkannte Fehler werden negativ gewertet!

```
#include <stdio.h>
#include <stdlib.h>
#define LEN 80;
char *to_reverse(char *str);
int main(int argc, char *argv[])
{
  char *str, *reverse;
  if(argc = 1)
    {
      fgets(str, LEN, stdin);
      str[strlen(str)-1]='\0'; /* \n am Ende löschen (diese Zeile stimmt so!) */
      reverse = to_reverse(str);
      printf("Normal: %s, Reverse: %s\n", str, reverse);
      free(reverse);
    }
  else
    {
      fprintf(stderr, "%s: Falscher Aufruf\n", &argv[0]);
      return -1;
    }
  return 0;
}
char *to_reverse(char *str)
{
  char *rev;
  int i, len;
  len = strlen(str);
  rev = (char *)malloc(len);
  for(i = 0; i<len; i++);
    rev[i] = str[len-i-1];
  rev[i] = '\0';

}
```

Aufgabe 3: (10 Punkte)

Was gibt das folgende Programm **`trace`** in Zeile 15 aus, wenn es wie folgt aufgerufen wird:

```
trace segmentationfault
```

Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an (für die erste Schleife in den **Zeilen 8/9** bitte nur die Werte des ersten und letzten Durchlaufs eintragen)! Verwenden Sie dazu die untenstehende Tabelle!

```
1   #include <stdio.h>
2   #define LEN 3
3   int main(int argc, char *argv[])
4   {
5     int i, len = 0;
6     char *str, result[LEN+1];
7     str = argv[1];
8     for(i=0; str[i]!='\0'; i++)
9       len++;
10    if(len > LEN)
11      len = LEN;
12    for(i=0; i<len; i++)
13      result[len-i-1] = str[i];
14    result[i] = '\0';
15    printf("%schafft\n", result);
16    return 0;
17  }
```

Zeile	i	len	str	result			
				result[0]	result[1]	result[2]	result[3]

Aufgabe 4: (5 Punkte)

Schreiben Sie eine **rekursive** Funktion, die den größten gemeinsamen Teiler zweier ganzer Zahlen berechnet. Der größte gemeinsame Teiler ist durch folgende Rekursionsgleichung definiert:

ggT(x,y) = y, wenn x modulo y gleich 0 ist,

ggT(x,y) = ggT(y, (x modulo y)), falls x modulo y ungleich 0 ist.

Aufgabe 5: (5 Punkte)

a) Welche Gefahr sehen Sie bei diesem Programmstück? Begründen Sie Ihre Antwort kurz.

```
float f, square;
for(f = 1.0; f != 0.0; f -= 0.1)
  {
    square = f * f;
    printf("Das Quadrat von %f ist %f\n", f, square);
  }
```

b) Was ist an folgender Programmzeile problematisch? Geben Sie ein Beispiel an, in dem das Problem auftritt!

```
#define QUADRAT(x) (x*x)
```