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
30	8	12	5	5	60

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

(Fachrichtung) (Semester)

Erlangen, 01.08.2000

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/result00.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, 01.08.2000

Aufgabe 1: (30 Punkte)

Schreiben Sie ein Programm **kunden** zum Erfassen einer beliebigen Anzahl von Kundendaten.

Für jeden Kunden sollen folgende Daten erfasst werden

- Nachname (max. 24 Buchstaben lang)
- Vorname (max. 24 Buchstaben lang)
- Geburtsdatum (10 Zeichen lang)
- Kundennummer (als Zahl).

Das Programm wird folgendermassen aufgerufen

kunde ***Ausgabedatei***

und arbeitet dann wie folgt beschrieben:

- Aufrufparameter wird ausgewertet, bei falschem Aufruf Fehlermeldung
- mit Hilfe der Funktion **`new_kunde`** wird eine Datenstruktur zur Speicherung der Kundendaten erzeugt, diese Struktur wird mit Hilfe der Funktion **`erfasse_daten`** gefüllt und vorne in eine verkettete Liste solcher Strukturen eingehängt. Dieser Schritt wird wiederholt, bis **`erfasse_daten`** das Ende der Eingabe meldet.
- schliesslich werden alle in der Liste gespeicherten Kundendaten mit Hilfe der Funktion **`speichere_liste`** 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.


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

/*
 * Deklaration der Kundenstruktur
*/
struct kunde {

      ________________________________

      ________________________________

      ________________________________

      ________________________________

      struct kunde *naechster;
};

/*
 * 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 kunde *new_kunde();
void speichere_liste(struct kunde *, 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 kunde *);
```

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

        ______________________________________________________
        ______________________________________________________
        ______________________________________________________

        /* Überprüfen der Kommandozeile */

        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________

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

        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________

        /*
         * Liste in Ausgabedatei speichern
         */

        ______________________________________________________

        return 0;
}
```



```
/*
 * Funktion new_kunde zum Erzeugen einer neuen Kundenstruktur
 * (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 kunde *new_kunde() {

        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________
        ______________________________________________________

}
```

```
/*
 * Funktion speichere_liste
 * 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
 * Nachname, Vorname, Geburtsdatum, Kundennummer
 * in die Datei geschrieben
 */
void speichere_liste(struct kunde *kunde, char *datei) {
      FILE *fp;

      /* Ausgabedatei öffnen */

      /* Liste ausgeben */

}
```


Aufgabe 2: (8 Punkte)

Markieren und verbessern Sie die Fehler (mind. 4, max. 7) in dem untenstehenden Programm, das zu gegebenem Jahr und Monatsnamen die Anzahl der Tage berechnet. Die Implementation der Funktion `schaltjahr` ist nicht angegeben. Sie liefert 1 wenn das Jahr ein Schaltjahr war, sonst 0. Beachten Sie, daß alle bis auf einen Fehler nicht als solche von Standard-Compilern erkannt werden.

Falsch erkannte Fehler werden negativ gewertet!

```
#include <stdio.h>
#include <string.h>
#define FEBRUAR 1;

static int tage[] = {31,28,31,30,31,30,31,31,30,31,30,31};
static char *monate[] = { "Januar","Februar","Maerz","April",
                "Mai", "Juni", "Juli", "August",
                "September", "Oktober", "November", "Dezember"};

int schaltjahr(int jahr);

int main()
{
  int i, ok = 0;
  int max_tage, jahr;
  char *monatsname;
  while(!ok);
    {
      printf("Monatsname: ");
      scanf("%32s", monatsname);
      printf("Jahr: " );
      scanf("%s", &jahr);
      for(i=1; i<=12; i++)
        {
          if(strcmp(monate[i], monatsname) == 0)
            {
              ok = 1;
              break;
            }
        }
    }
  max_tage = tage[i];
  if(i = FEBRUAR)
    if(schaltjahr(jahr))
      max_tage = 29;
  printf("%s %d hat %d Tage\n", monatsname, jahr, max_tage);
  return 0;
}
```

Aufgabe 3: (12 Punkte)

Was gibt das folgende Programm in Zeile 7 aus? Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an! Verwenden Sie dazu die untenstehende Tabelle!

```
1   #include <stdio.h>
2   int s(int a[], int b);
3   int main()
4   {
5     int i, a[] = { 2, 7, 5 };
6     i = s(a, 3);
7     printf("%d mal => %d %d %d\n", i, a[0], a[1], a[2]);
8     return 0;
9   }
10  int s(int a[], int b)
11  {
12    int i, t, f = 0;
13    for(i=0; i<b-1; i++)
14      if(a[i]>a[i+1])
15        {
16          t = a[i];
17          a[i] = a[i+1];
18          a[i+1] = t;
19          f++;
20        }
21    return f;
22  }
```

Zeile	i_{main}	a_{main} a[0] a[1] a[2]	a_s	b_s	i_s	t_s	f_s

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 = 0.0;
while(1)
  {
    f = f + 0.1,
    if(f == 1.0)
      break;
  }
```

b) Was ist der Unterschied zwischen einer Funktionsdefinition und einer Funktionsdeklaration (Prototyp)? Warum werden Funktionsdeklarationen benötigt?