Friedrich-Alexander-Universität Erlangen-Nürnberg

Schriftliche Prüfung zu der Lehrveranstaltung
„Grundlagen der Informatik II — Systemnahe Programmierung in C (SPiC)“
im Studiengang Elektrotechnik, Elektronik und Informationstechnik

	erreichbare Punkte	erhaltene Punkte						
Aufgabe 1	**14**							
Aufgabe 2	**30**	**2a**	**2b**					
Aufgabe 3	**16**	**3a**	**3b**	**3c**	**3d**	**3e**	**3f**	
Summe	**60**							
Note								

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

(Studiengang) (Semester)

Durch meine Unterschrift bestätige ich
- den Empfang der vollständigen Klausurunterlagen (11 Seiten inkl. Deckblatt),
- die Kenntnisnahme der Hinweise auf Seite 2.

Erlangen, 09.04.2008 (Unterschrift)

Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung auf Seite 1 unterschreiben.

- Die Bearbeitungszeit beträgt 60 Minuten.
- Es sind **keine** eigenen Hilfsmittel zugelassen.
- Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen Raum geschrieben werden. Beachten Sie bitte, dass der freigelassene Platz großzügig bemessen ist und nicht unbedingt der erwarteten Antwortlänge entspricht. Sollte der Platz nicht ausreichen, können Sie die Rückseiten der Aufgabenblätter mitverwenden. Kennzeichnen Sie dabei die Zugehörigkeit Ihrer Lösung zu einer Aufgabe. Bei Bedarf können zusätzliche Lösungsblätter (weiß) ausgeteilt werden. Vermerken Sie vor deren Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf!
- Die Lösungen müssen dokumentenecht in blau oder schwarz geschrieben werden. Als falsch Erkanntes muss deutlich durchgestrichen werden. Tintenkiller darf nicht verwendet werden. Keinen Bleistift verwenden!
- Schmierpapier darf **nicht** abgegeben werden. Bei Bedarf ist von der Aufsicht weiteres Schmierpapier (farbig) erhältlich.
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studiengang 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.
- In den letzten 15 Minuten der Bearbeitungszeit können Sie den Raum nicht mehr verlassen. Bleiben Sie an Ihrem Platz sitzen, bis am Ende alle Klausurunterlagen eingesammelt und nachgezählt sind und die Aufsicht das Zeichen zum Gehen gibt.

Die **Klausurergebnisse** werden in ca. zwei Wochen im WWW unter der Seite der Vorlesung im SS 2007, Unterpunkt "Ergebnisse" veröffentlicht.

Aufgabe 1: (14 Punkte)

Bei den Multiple-Choice-Fragen ist jeweils nur **eine** richtige Antwort eindeutig anzukreuzen. Auf die richtige Antwort gibt es die angegebene Punktzahl.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, kreisen Sie bitte die falsche Antwort ein und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Der Speicher eines Prozesses ist grob in die Bereiche Code, Daten und Stack aufgeteilt. Wozu dient die Unterscheidung von Daten-Bereich und Stack-Bereich? 2 Punkte

- ❏ Variablen, deren Speicher für die gesamte Programmlaufzeit verfügbar sein muss, liegen im Daten-Bereich. Lokale Variablen, deren Speicher nur während der Ausführung einer Funktion bereit steht, liegen im Stack.
- ❏ Alle Variablen des Programms werden im Datenbereich angelegt. Sollte dieser nicht ausreichen, wird zusätzlich der Stack hergenommen.
- ❏ Die Variablen des Programms liegen im Stack, Felder liegen im Daten-Bereich.
- ❏ Daten- und Stack-Bereich sind eigentlich das Gleiche, die unterschiedlichen Namen stammen aus der Linux- bzw. Windows-Welt.

b) Welche Aussage zu Zeichenketten in C (C-Strings) ist richtig? 2 Punkte

- ❏ C-Strings speichern im ersten Byte ihre Länge.
- ❏ C-Strings werden in C durch den Datentyp `String` repräsentiert.
- ❏ C-Strings werden durch ein `char`-Feld implementiert.
- ❏ C-Strings können maximal 255 Zeichen enthalten.

c) Welche der folgenden Aussagen über den C-Präprozessor ist richtig? 2 Punkte

- ❏ Der Präprozessor ist eine Hardwarekomponente, die den Hauptprozessor beim Übersetzen von C-Code unterstützt.
- ❏ Der Präprozessor expandiert Makros durch textuelle Ersetzung.
- ❏ Nach dem Übersetzen und dem Binden müssen C-Programme durch den Präprozessor nachbearbeitet werden, um Makros aufzulösen.
- ❏ Der Präprozessor ist Teil des Betriebssystemkerns.

d) Was ist der Unterschied zwischen den wie folgt in der Datei prog.c global definierten Variablen? 2 Punkte

```
int a = 2;
static int b = 3;
```

- ❏ Die Variable a ist nur für Funktionen in der Datei prog.c zugreifbar während auf b auch von anderen Modulen des Programms erreichbar ist, wenn es dort als extern deklariert wird.
- ❏ Der Speicherplatz der Variablen a wird jeweils beim Aufruf einer Funktion angelegt und beim Verlassen wieder freigegeben während der Speicherplatz der Variablen b von Programmstart bis -ende verfügbar ist.
- ❏ Die Variable b ist nur für Funktionen in der Datei prog.c zugreifbar. Funktionen in anderen Modulen des Programms können auf a zugreifen wenn in dem entsprechenden Modul a mit einer extern-Deklaration bekannt gemacht wurde.
- ❏ Der Wert der Variablen b ist konstant und kann sich während der Ausführung des Programms nicht ändern. a kann dagegen beliebig verändert werden.

e) Was versteht man unter formalen und aktuellen Parametern einer Funktion? 2 Punkte

- ❏ Der aktuelle Parameter enthält eine Kopie des Aufrufparameters, der formale Parameter ist ein Zeiger auf den Aufrufparameter.
- ❏ Der formale Parameter ist der Name, unter dem auf einen Funktionsparameter innerhalb der Funktion zugegeriffen werden kann. Der aktuelle Parameter ist der beim tatsächlichen Funktionsaufruf übergebene Wert.
- ❏ Die formalen Parameter sind die Aufrufparameter einer Funktion, die aktuellen Parameter sind die Rückgabewerte.
- ❏ Die formalen Parameter beschreiben die Typen der Funktionsparameter, die aktuellen Parameter sind die Namen unter denen innerhalb der Funktion auf die Parameter zugegriffen wird.

f) Welcher der folgenden Mechanismen gehört **nicht** zu den Funktionen eines Betriebssystemkerns wie z. B. UNIX oder Windows? 2 Punkte

- ❏ Verwaltung des Hauptspeichers
- ❏ Dateisystem
- ❏ Interrupt-Behandlung
- ❏ Compiler

g) Was bewirkt folgende Funktion?

2 Punkte

```
void func(int *a, int *b) {
  int c = *a; *a = *b; *b = c;
}
```

- ❏ Bei einem Aufruf vertauscht die Funktion die Inhalte der von a und b adressierten Speicherzellen.
- ❏ Da in C Funktionen mit "call by value" aufgerufen werden, erhält die Funktion Kopien der Aufrufparameter, die sie vertauscht. Beim Aufrufer hat dies allerdings keine Auswirkungen.
- ❏ Der Compiler meldet bei der Übersetzung einen Fehler, weil `*a` nicht auf der linken Seite eines =-Zeichens stehen darf.
- ❏ Die von b adressierte Speicherzelle enthält nach dem Aufruf die in der Variablen a abgelegte Speicheradresse.

Aufgabe 2a: (10 Punkte)

Ein Mikrocontroller steuert eine Beleuchtungsanlage, die per Tastendruck ein- und ausgeschaltet werden kann, über ein an Adresse 0x3B in den Speicher eingeblendetes 8-Bit Steuerregister.

Der Zustand des Tasters kann über Pin 0 dieses Registers abgefragt werden (0=nicht gedrückt, 1=gedrückt), die Beleuchtungsanlage wird über Pin 1 gesteuert (0=ausgeschaltet, 1=eingeschaltet).

Schreiben Sie ein Steuerprogramm, welches durch Polling den Zustand des Tasters fortlaufend abfragt. Bei jedem Druck des Tasters soll der Zustand der Beleuchtung umgeschaltet werden. Achten Sie darauf, längeres Drücken des Tasters nicht mehrfach zu interpretieren.

```
 /* Makro fuer Zugriff auf das Register */
```

```
/* Funktion main */
```

C:

Aufgabe 2b: (20 Punkte)

Schreiben Sie eine Kontrollfunktion

```
int kontrolle(int werte[], unsigned int anzahl, int grenzwert);
```

welche in einem Feld `werte` von `anzahl` Messwerten nach Überschreitungen des Grenzwertes `grenzwert` sucht. Übersteigt die Zahl der Grenzwertüberschreitungen in einem Aufruf die Schwelle 3, so soll die Funktion

```
void alarm(int alarmwerte[], unsigned int anzahl, int grenzwert);
```

aufgerufen werden. Dieser soll ein Feld `alarmwerte` mit allen Werten, welche den Grenzwert überschreiten, übergeben werden, zusammen mit der Anzahl `anzahl` dieser Werte und dem Grenzwert `grenzwert` selbst.

- Das Feld mit den Alarmwerten muss von ihrer Kontrollfunktion dynamisch allokiert werden.
  Hierzu steht Ihnen die Funktion malloc zur Verfügung.
  Schnittstelle: `void *malloc(size_t size);`
- Vor dem Rücksprung muss Ihre Kontrollfunktion das allokierte Feld wieder freigeben.
  Funktion `free`, Schnittstelle: `void free(void *ptr);`
- Die Funktion soll die Zahl der gefundenen Grenzwertüberschreitungen zurückliefern, oder `-1`, falls bei der Bearbeitung ein Fehler auftritt.
- Sie können davon ausgehen, dass die Kontrollfunktion nur mit sinnvollen Parametern aufgerufen wird.

Implementieren Sie ausserdem die oben aufgerufene Alarmfunktion. Diese soll die Meldung "`x Werte ueberschreiten den Grenzwert:`" auf die Standardausgabe ausgeben, wobei `x` die Zahl der überschrittenen Grenzwerte darstellt. Anschliessend soll jeder dieser Werte in einer eigenen Zeile auf der Standardausgabe ausgegeben werden.

Ergänzen Sie das folgende Codegerüst so, dass ein vollständig übersetzbares Programm entsteht.

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

/* Makros, Funktiondeklarationen, etc. */

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

/* Funktion kontrolle */

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
    /* Alarmwerte Feld allokieren */

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
   /* Messwerte untersuchen */

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
```

```
   /* Alarmfunktion aufrufen, falls notwendig */
```

```
   /* Alarmwerte-Feld freigeben */
```

```
/* Ende der Funktion kontrolle */

/* Funktion alarm */
```

```
/* Ende der Funktion alarm */
```

K:A:

Aufgabe 3: (16 Punkte)

Die folgenden Beschreibungen sollen kurz und prägnant erfolgen (Stichworte, kurze Sätze)

Es gibt zwei verschiedene Möglichkeiten, wie ein Programm mit Ereignissen an Peripherie-Schnittstellen umgehen kann: Polling und Interrupt-Verarbeitung.

a) Warum könnte ein externes Gerät die "Aufmerksamkeit" der CPU / des Programmes wünschen? Nennen Sie 3 Beispiele.

b) Beschreiben Sie die beiden Verfahren (Polling und Interrupts) in Stichpunkten .

c) Was versteht man unter Nebenläufigkeit?

d) In welchen Situation tritt Nebenläufigkeit auf?

e) Welches grundsätzliche Problem entsteht durch Nebenläufigkeit?
Beschreiben Sie das Problem ausserdem anhand eines konkreten Beispiels.