Aufgabe 1: (19 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) Gegeben sei folgende Deklaration in einem C-Programm: 2 Punkte

```
typedef void (*func)(int, int);
```

Welche Aussage ist **richtig**?

- ❏ `func` ist ein Funktionszeiger auf eine `void`-Funktion, die zwei `int`-Parameter erwartet.
- ❏ `func` ist Datentyp für Funktionszeiger auf `void`-Funktionen, die zwei `int`-Parameter erwarten.
- ❏ `func` ist eine Funktion, die hier durch forward-Deklaration dem Compiler bekannt gegeben wird.
- ❏ Der Compiler meldet einen Fehler, weil bei der Deklaration von `func` die Namen der formalen Parameter fehlen.

b) Gegeben sei folgende Enumeration: 2 Punkte

```
enum SPRACHE {Deutsch, Englisch, Russisch};
```

Welche Aussage ist **richtig**?

- ❏ Der Compiler meldet einen Fehler, weil den enum-Elementen kein Wert zugewiesen wurde.
- ❏ Der Wert von Russisch ist 3.
- ❏ Der Wert von Russisch ist 2.
- ❏ Der Wert von Russisch ist unbekannt; der Compiler weist zur Übersetzungszeit jedem enum-Element einen zufälligen, aber eindeutigen Wert zu.

c) In welchen Situationen wird ein Prozess in den Zustand "blockiert" versetzt? 2 Punkte

- ❏ während der Prozesserzeugung solange die Verwaltungsstrukturen noch nicht angelegt sind
- ❏ bei jedem Aufruf der V-Operation eines Semaphors
- ❏ wenn er ausgeführt werden könnte, aber ein anderer Prozess die CPU zugeteilt bekommen soll
- ❏ beim Lesen eines Zeichens von der Tastatur, so lange keine Taste gedrückt wird

d) Welche der folgenden Aussagen bzgl. der Interruptsteuerung ist **richtig**? 2 Punkte

- ❏ Wurde gerade ein Pegel-gesteuerter Interrupt ausgelöst, so muss erst ein Pegelwechsel der Interruptleitung stattfinden, bevor erneut ein Interrupt ausgelöst wird.
- ❏ Während der Bearbeitung eines Interrupts nimmt der Prozessor keine weiteren Interrupts an.
- ❏ Pegel-gesteuerte Interrupts werden beim Wechsel des Pegels ausgelöst.
- ❏ Flanken-gesteuerten Interrupts können nicht blockiert werden, da sie völlig unvorhersehbar auftreten.

e) Welche der folgenden Aussagen zum Binden eines Programms ist **richtig**? 2 Punkte

- ❏ Auf einer Mikrocontroller-Plattform wird normalerweise dynamisch gebunden, weil dies Speicherplatz spart.
- ❏ Der Binder löst #include-Anweisungen in einem C-Programm auf, indem er sie durch den Inhalt der einzubindenden Datei ersetzt.
- ❏ Der Binder fügt Objektdateien und Bibliotheken zu einer ausführbaren Datei zusammen.
- ❏ Beim Binden werden Zugriffe aus einem C-Modul auf globale static-Variablen eines anderen C-Moduls optimiert.

f) Folgende Makrodefinition findet sich in der AVR-Bibliothek: 2 Punkte

```
#define PINA (*(volatile uint8_t *)0x39)
```

Welche der folgenden Aussagen bezüglich der Verwendung des `volatile`-Schlüsselworts ist in diesem Fall **richtig**?

- ❏ Das `volatile`-Schlüsselwort stellt hier sicher, dass der Zugriff auf `PINA` mit Interrupts synchronisiert wird.
- ❏ Auf der AVR-Plattform werden Hardware-Register (wie `PINA`) im RAM eingeblendet. Das `volatile`-Schlüsselwort zeigt an, dass es sich dabei um flüchtigen Speicher handelt.
- ❏ Wird der Port A als Eingang konfiguriert, könnte sich der Wert von `PINA` jederzeit ändern. Durch `volatile` wird der Compiler angewiesen, stets den aktuellen Wert aus `PINA` zu lesen.
- ❏ Das `volatile`-Schlüsselwort ermöglicht den sicheren Zugriff auf einzelne Bits des Registers.

g) Gegeben sei folgendes Programmfragment für einen AVR-Microcontroller: 3 Punkte

```
uint8_t a = 100;
uint8_t b;

b = a+a * 2-50;
```

Welche der folgenden Aussagen ist **richtig**?

- ❏ b hat nach Ausführung der Zuweisung den Wert 250.
- ❏ b hat nach Ausführung der Zuweisung den Wert 350.
- ❏ Der Compiler warnt zur Übersetzungszeit vor einem möglichen Bereichsüberlauf.
- ❏ Während der Ausführung kommt es zu einem Bereichsüberlauf; auf der AVR-Plattform bleibt dieser jedoch unentdeckt.

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

- ❏ Die formalen Parameter sind die Aufrufparameter einer Funktion, die aktuellen Parameter sind die Rückgabewerte.
- ❏ Der aktuelle Parameter enthält eine Kopie des Aufrufparameters, der formale Parameter ist ein Zeiger auf den Aufrufparameter.
- ❏ Die formalen Parameter beschreiben die Typen der Funktionsparameter, die aktuellen Parameter sind die Namen, unter denen innerhalb der Funktion auf die Parameter zugegriffen wird.
- ❏ 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.

i) Welche der folgenden Aussagen zum Thema Threads ist **richtig**? 2 Punkte

- ❏ Kernel-Level-Threads können blockieren, ohne andere Threads zu behindern.
- ❏ User-Level-Threads sind die effizienteste Möglichkeit ein Multiprozessorsystem zu nutzen.
- ❏ Kernel-Level-Threads dürfen in normalen Anwendungsprogrammen aus Sicherheitsgründen nicht verwendet werden.
- ❏ Im Gegensatz zu User-Level-Threads läuft jeder Kernel-Level-Thread in einem eigenen Adressraum.

Aufgabe 2a: Konfigurierbarer Würfel (30 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie ein Würfelprogramm für den AVR-Mikrocontroller, welches die gewürfelten Zahlen auf einem LED-Feld anzeigt. Der Würfel soll eine Zahl im Intervall [1; n] anzeigen, wobei der Höchstwert n durch den Benutzer festgelegt werden soll (Maximum: 8, Anfangswert: 1). Ihr Programm soll sich in zwei Phasen gliedern:

Phase 1: Initialisierung des Höchstwerts

- In einer Schleife wird der Wert n jeweils um 1 hochgezählt (nach der 8 folgt wieder die 1); gleichzeitig wird der Wert n durch den Füllstand der LEDs angezeigt (Beispiel: n == 3, LEDs 1-3 leuchten).
- Zwischen den Inkrementschritten wartet das Programm für 1000 Schleifendurchläufe in einer aktiven Warteschleife (Funktion `void wait(void);`).
- Durch Drücken von BUTTON0 wird die Initialisierung abgeschlossen; der aktuelle Wert n wird dadurch als Höchstwert für die anschließende Würfelphase festgelegt.

Phase 2: Würfelphase

- In der Würfelschleife soll das Programm die Zahlen im Intervall [1; Höchstwert] direkt nacheinander auf den LEDs darstellen (ohne Wartezeit, dadurch für das Auge kaum erkennbar). Eine Zahl wird dadurch dargestellt, dass nur die entsprechende LED angeschaltet wird; die restlichen LEDs sind ausgeschaltet.
- Wenn der Benutzer nun BUTTON0 drückt, hält das Programm an und die gerade leuchtende LED gilt als gewürfelt. Das Programm wartet auf einen weiteren Tastendruck (Schlafmodus!).
- Durch einen erneuten Druck auf BUTTON0 wird die Würfelschleife wieder fortgesetzt, bis durch einen weiteren Druck auf BUTTON0 wieder angehalten wird. Dies wird endlos so fortgesetzt.

Informationen zur Hardware und Bibliothek:

- Zur LED-Ansteuerung stehen Ihnen keine Bibliotheksfunktionen zur Verfügung. Die 8 LEDs hängen an Port C und sind active low. Zur Konfiguration als Output-Pin müssen Sie das entsprechende Bit im DDRC-Register auf 1 setzen; zur Ausgabe eines High-Pegels müssen Sie das entsprechende Bit im PORTC-Register auf 1 setzen.
- Der Taster ist hardwareseitig entprellt. Um eine Callback-Funktion für BUTTON0 zu registrieren, gehen Sie davon aus, dass Ihnen die folgende Bibliotheksfunktion zur Verfügung steht:

  ```
  void registerCallbackButton0(void (*callback)(void));
  ```

  Als Parameter erhält diese Funktion den Namen Ihrer Callback-Funktion, die dann automatisch aus dem Interrupthandler aufgerufen wird, wenn der Taster gedrückt wird.
- Programmieren Sie die Hardwareinitialisierung in einer Funktion

  ```
  void init(void);
  ```

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

```
#include <avr/io.h>
#include <avr/interrupt.h>

/* Funktiondeklarationen, globale Variablen, etc. */

.....
.....
.....
.....
.....
.....
.....
.....
/* Funktion main */
.....
.....
.....
.....
  /* Aufruf der Hardware-Initialisierung
     und Initialisierung der Callback-Funktion */
.....
.....
.....
.....
.....
.....
.....
```

A:


```
    /* Initialisierungsphase */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
    /* Wuerfelphase */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
/* Ende der main-Funktion */
```

L:

```
/* Initialisierungsfunktion */
```

```
/* Wartefunktion */
```

```
/* Callbackfunktion fuer BUTTON0 */
```

F:

Aufgabe 2b: backuplist (15 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie ein Programm **`backuplist`**, das für Zwecke der Datensicherung eine Liste aller seit der letzten Datensicherung veränderten regulären Dateien ausgibt.

Das Programm bearbeitet jeweils genau ein Verzeichnis, der komplette Pfadname des Verzeichnisses wird beim Aufruf von backuplist als Argument übergeben.

Der Zeitpunkt der letzten Datensicherung ist anhand der Modifikationszeit (**`st_mtime`**) der Datei **`backup.timestamp`** in dem Verzeichnis ersichtlich.

Das Programm **`backuplist`** soll die vollständigen Pfadnamen aller zu sichernden Dateien der Standardausgabe ausgeben.

Hinweis:

Auf Daten vom Typ **`time_t`** können normale Vergleichsoperationen durchgeführt werden.

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

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/stat.h>
#include <unistd.h>

/* Funktion main */
```

```
/* Aufrufargument pruefen, Referenzzeitpunkt ermitteln */
```


```
/* Verzeichnis oeffnen und durchsuchen, Dateinamen ausgeben */
```

```
/* Ende der Funktion main */
```

B:

Aufgabe 3: (15 Punkte)

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

a) Beschreiben Sie die Unterschiede bei der Ausführung eines Programms auf einem Mikrokontroller ohne Betriebssystem und auf einem Betriebssystem wie z. B. Linux. (8 Punkte)

b) Welche Bedeutung haben die unterschiedlichen Gültigkeitsbereiche von Variablen in einem C-Programm in Bezug auf Nebenläufigkeit bei Interrupts (d. h. wie sind Variablen der verschiedenen Kategorien von Nebenläufigkeitsproblemen betroffen und warum ist dies so)? (4 Punkte)

c) Gegeben sei folgendes Programmfragment:

```
union {
    struct {
        uint8_t reg_lo, reg_hi;
    };
    uint16_t reg16;
} reg;
```

Wieviel Speicher (in Bytes) wird durch die Variable reg belegt? Skizzieren Sie außerdem, wie die einzelnen Komponenten von reg im Speicher abgelegt werden. (3 Punkte)

Aufgabe 4: (11 Punkte)

a) Bei Ausnahmesituationen, wie sie bei der Ausführung eines Programms auftreten können, werden ***Traps*** und ***Interrupts*** unterschieden.

Beschreiben Sie diese beiden Arten:
i) Wodurch werden diese Ausnahmesituationen ausgelöst?
ii) Wodurch unterscheiden sie sich?
iii) Geben Sie jeweils ein Beispiel an.
(6 Punkte)

b) Beschreiben Sie den Ablauf einer typischen Interruptverarbeitung auf einem Mikrocontroller (5 Punkte)