Aufgabe 1: (25 % der Klausurnote)

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, streichen Sie bitte die falsche Antwort mit drei waagrechten Strichen durch (☒)und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Welche Aussage zu Zeigern ist richtig? ☐

- ❏ Der Speicherbedarf eines Zeigers ist abhängig von der Größe des Objekts, auf das er zeigt.
- ❏ Zeiger können verwendet werden, um in C eine call-by-reference Übergabesemantik nachzubilden.
- ❏ Die Übergabesemantik für Zeiger als Funktionsparameter ist call-by-reference.
- ❏ Zeiger vom Typ `(void *)` existieren in C nicht, da solche "Zeiger auf Nichts" keinen sinnvollen Einsatzzweck hätten.

b) Was bewirken folgende Programmanweisungen? ☐

```
uint8_t x = 42;
x ^= x;
```

- ❏ Alle Bits der Variable ändern ihren Wert.
- ❏ Alle Bitwerte werden um eine Stelle nach links verschoben.
- ❏ Die Variable hat nach der Operation den Wert 0.
- ❏ Die Variable hat nach der Operation den Wert -42.

c) Wann kann es zu Nebenläufigkeitsproblemen kommen? ☐

- ❏ Wenn die Programmabschnitte im if- und else-Teil einer bedingten Anweisung auf diesselbe Variable zugreifen.
- ❏ Wenn ein Programm in einer Funktion mehrere lokale Variablen verwendet.
- ❏ Wenn aus dem Hauptprogramm und einer Unterbrechungsbehandlungsfunktion dieselbe globale Variable geschrieben wird.
- ❏ Wenn ein Programmabschnitt in einer Schleife mehrfach durchlaufen wird.

d) Was ist der Unterschied zwischen den wie folgt in einer C-Datei global definierten Variablen?

```
int a;
static int b;
```

- ❏ Variable b kann nur einmalig zugewiesen werden und bleibt danach konstant.
- ❏ Die Variable b ist nur für Funktionen in derselben Datei zugreifbar, während auf Variable a auch von Funktionen in anderen Modulen des Programms zugegriffen werden kann.
- ❏ Die Variable b ist nur zugreifbar aus Funktionen, die ebenfalls mit dem Schlüsselwort **`static`** deklariert wurden.
- ❏ Die Variable a ist nur für Funktionen in derselben Datei zugreifbar, während auf Variable b auch von Funktionen in anderen Modulen des Programms zugegriffen werden kann.

e) Welche der folgenden Aussagen zum Thema Threads ist richtig?

- ❏ 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.

f) Wie viele Bytes belegt die folgende Struktur im Speicher eines AVR-Mikrocontrollers:

```
union {
    struct {
        uint8_t lo, hi;
    };
    uint16_t r16;
} reg;
```

- ❏ 2 Bytes
- ❏ 4 Bytes
- ❏ 16 Bytes
- ❏ 32 Bytes

g) Welche Aufgabe erfüllt der C-Präprozessor?

- ❏ Er bindet mehrere .o-Dateien zusammen.
- ❏ Er löst beim Binden die Referenzen zwischen Programm und Bibliotheken auf.
- ❏ Er entfernt vor dem Kompilieren ungenutzte Variablen aus dem Programm.
- ❏ Er führt textuelle Ersetzungen im C-Code durch, die sich durch Makrodefinitionen steuern lassen.

h) Welche Aussage zu Variablen in C ist richtig?

- ❏ Variablen vom Typ `int` haben auf allen Hardware-Plattformen dieselbe Größe.
- ❏ Eine `int8_t`-Variable braucht mehr Speicher als eine `uint8_t`-Variable, weil zusätzlicher Platz für das Vorzeichen-Bit benötigt wird.
- ❏ Alle Variablen werden bei ihrer Deklaration mit dem Wert 0 initialisiert.
- ❏ Wenn man bei einer `int8_t`-Variable zum Wert 127 eins dazu addiert, hat sie den Wert -128.

i) Was versteht man unter Polling?

- ❏ Wenn ein Programm regelmäßig eine Peripherie-Schnittstelle überprüft, ob Daten oder Zustandsänderungen vorliegen.
- ❏ Das regelmäßige Anheben eines Pegels, um einem Gerät einen bestimmten Zustand zu signalisieren.
- ❏ Wenn ein Programm zum Zugriff auf kritische Daten Interrupts sperrt.
- ❏ Wenn gleichzeitig mehrere Interrupt-Requests anliegen.

j) Welche Aussage zu Semaphoren ist richtig?

- ❏ Eine P-Operation überprüft, ob der Semaphor den Wert 0 hat und erhöht ihn anschließend.
- ❏ Mit Hilfe der P-Operation kann sichergestellt werden, dass ein kritischer Abschnitt nicht von zwei Prozessen gleichzeitig betreten wird.
- ❏ Die P-Operation wird benutzt, um in kritischen Abschnitten Interrupts zu sperren.
- ❏ Die V-Operation dekrementiert den Semaphor um 1 und deblockiert andere in einer V-Operation wartende Prozesse.

Aufgabe 2a: gate (31 Punkte)

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

Schreiben Sie eine Steuerung für einen AVR-Mikrocontroller, der den Antrieb eines Garagentors kontrolliert. Mit einem Kippschalter (Stellungen "hoch" und "runter") kann das Tor bewegt werden. Erreicht das Tor den geöffneten oder geschlossenen Zustand, soll der Motor abgeschaltet werden. Während der Bewegung des Tors zeigt eine vertikale Lichtleiste aus acht LEDs die aktuelle Position des Tors an, indem zu jedem Zeitpunkt genau eine LED aktiv ist.

Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion **`void init(void);`**. Es sollen keine Annahmen über den initialen Zustand der Hardware-Register getroffen werden.
- Das Programm startet mit geschlossenem Tor, alle LEDs und der Motor sind aus, der Kippschalter steht auf "runter".
- Während des Wartens auf den Kippschalter soll der Mikrocontroller in den Schlafmodus gehen. Eine Bewegung des Kippschalters löst einen Interrupt aus und der Motor soll mit der entsprechenden Drehrichtung eingeschaltet werden.
- Zur Überwachung der Bewegung wird, während der Motor läuft, alle 100 ms die Position des Tors abgefragt und auf der LED-Leiste angezeigt. Die aktuelle Position liefert ein Sensor an einem Analog-Digital-Umsetzer (ADC), dessen Wert mit der Bibliotheksfunktion **`uint16_t adc_read(void);`** abgefragt werden kann. Die Funktion liefert einen 10-Bit-Wert zurück (0 = offen, 1023 = geschlossen). Dieser Wertebereich soll gleichmäßig auf die acht LEDs abgebildet werden.
- Implementieren Sie das periodische Abfragen des Sensors unter Verwendung einer aktiven Wartefunktion **`void wait(uint16_t ms);`** , die **`ms`** Millisekunden wartet. Ihnen steht eine Präprozessorkonstante **`LOOPS_PER_MS`** zur Verfügung, die angibt, wieviele Schleifendurchläufe einer Millisekunde entsprechen.
- Wird eine der Endpositionen erreicht, soll der Motor abgeschalten werden. Ein Wechsel der Drehrichtung des Motors soll jederzeit möglich sein.

Information über die Hardware

LEDs: **PORTC**, Pins 0-7, Start bei LED 1 an Pin 0, eingeschaltet bei low-Pegel
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRC**-Register auf 1

Motor: **PORTA**, Pins 0 und 1
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRA**-Reg. auf 1
- Wahl der Drehrichtung an Pin 0 (0 runter, 1 hoch)
- Aktivierung des Motors an Pin 1 (0 aus, 1 an)

Kippschalter: **PORTD**, Interrupt-Leitung an Pin 2, Schalter auf "Runter" = Pin 3, "Hoch" = Pin 4
- Pin als Eingang konfigurieren: entsprechendes Bit in **DDRD**-Register auf 0
- externe Interruptquelle **INT0**, ISR-Vektor-Makros: **INT0_vect**
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT0**-Bits im Register **GICR**.
- die Interrupt-Leitung verbindet den Pin mit Masse, es muss der interne Pullup-Widerstand verwendet werden (entsprechendes Bit in **PORTD**-Register auf 1 setzen).
- Konfiguration der externen Interruptquelle 0 (Bits in Register **MCUCR**)

ISC01	ISC00	Beschreibung
0	0	Interrupt bei low Pegel
0	1	Interrupt bei beliebiger Flanke
1	0	Interrupt bei fallender Flanke
1	1	Interrupt bei steigender Flanke

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

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

#define LOOPS_PER_MS 200
#define RUNTER 1
#define HOCH   2

uint16_t adc_read(void);
/* Funktionsdeklarationen, globale Variablen, etc. */

/* Unterbrechungsbehandlungsfunktion */
```

A:

```
/* Funktion main */

   /* Initialisierung */

   /* Hauptschleife */

    /* warten auf Schalterbewegung */
```

```
        /* Motor anschalten, Bewegung ueberwachen */
```

```
    /* Motor ausschalten */
```

```
/* Ende main */
```

M:

```
/* Initialisierungsfunktion */
```

```
/* Wartefunktion */
```

IW:

Aufgabe 2b: (14 Punkte)

Schreiben Sie eine Funktion **`void compiledir()`**, welche alle C-Dateien (Dateiendung ".c") im aktuellen Verzeichnis übersetzt.

Die Funktion soll wie folgt funktionieren:

- Sämtliche Fehlermeldungen sollen auf den Standardfehlerkanal `stderr` ausgegeben werden, nicht auf die Standardausgabe. Im Fehlerfall soll die Funktion das gesamte Programm beenden.
- Die Funktion durchsucht das aktuelle Verzeichnis. Für alle gefundenen Dateien, die mit den Zeichen ".c" enden, wird die Funktion

  ```
  void compile(const char *datei);
  ```

  aufgerufen. Dieser Funktion wird der Name der C-Datei übergeben.
  Die Funktion **`compile`** selbst soll nicht programmiert werden!
- Die Funktion **`compiledir`** kehrt zurück, nachdem das Verzeichnis vollständig durchlaufen wurde.

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

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>
#include <errno.h>
#include <unistd.h>
/* Funktiondeklarationen, globale Variablen, etc. */

----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
/* Funktion compiledir */

----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
/* aktuelles Directory oeffnen */

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

```
  /* Directory durchsuchen und .c-Dateien compilieren */
```

```
  /* Ressourcen freigeben */
```

```
/* Ende der Funktion compiledir */
```

D:

Aufgabe 3: (6 Punkte)

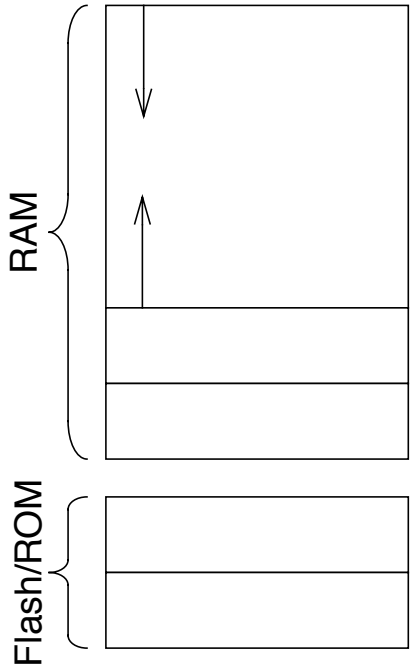
Die folgenden Beschreibungen sollen kurz und prägnant erfolgen (Stichworte, kurze Sätze)

a) Was versteht man unter einem Interrupt? (3 Punkte)

b) In welchen Schritten wird ein Interrupt typischerweise auf einem Mikrocontroller bearbeitet? (3 Punkte)

Aufgabe 4: (8 Punkte)

a) Beschreiben Sie die auf einem AVR-Mikrocontroller vorkommenden Speicherbereiche und zeichnen Sie diese in die Grafik ein. (5 Punkte)



b) Erklären Sie die Unterschiede zwischen Flash/ROM und RAM. (1 Punkt)

c) Was passiert mit den verschiedenen Speicherbereichen beim Systemstart? (2 Punkte)

Aufgabe 5: (9 Punkte)

a) Was versteht man unter Nebenläufigkeit? Geben Sie eine kurze Definition des Begriffs. (2 Punkte)

b) In welchen Situationen tritt Nebenläufigkeit auf? (2 Punkte)

c) Bei Nebenläufigkeit können verschiedene Probleme auftreten. Beschreiben Sie exemplarisch zwei solche Probleme und wie man mit ihnen umgeht, um korrekt funktionierende Software zu gewährleisten. (5 Punkte)