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, 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? 2 Punkte

- ❑ Ein Zeiger darf nie auf seine eigene Speicheradresse verweisen.
- ❑ Der Speicherbedarf eines Zeigers ist unabhängig von der Größe des Objekts, auf das er zeigt.
- ❑ Ein Zeiger kann zur Manipulation von schreibgeschützten Datenbereichen verwendet werden.
- ❑ Zeiger vom Typ `(void *)` sind am besten für Zeigerarithmetik geeignet, da sie kompatibel zu jedem Zeigertyp sind.

b) Gegeben ist folgender Programmcode: 2 Punkte

```
#define SUB(a,b) a-b
#define MUL(a,b) a*b
```

Was ist das Ergebnis von folgendem Ausdruck:

```
4 * SUB( MUL(2,3), 4)
```

- ❑ -8
- ❑ 8
- ❑ 20
- ❑ 96

c) 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.
- ❑ Beim Auftreten eines Interrupts sichert der Prozessor automatisch Register in den Speicher.
- ❑ Pegel-gesteuerte Interrupts werden beim Wechsel des Pegels ausgelöst.
- ❑ Flanken-gesteuerten Interrupts können nicht blockiert werden, da sie völlig unvorhersehbar auftreten.

d) Welche Aussage zu *volatile* ist richtig? 2 Punkte

- ❏ Das Schlüsselwort *volatile* unterbindet Optimierungen an einer damit definierten Variable.
- ❏ Das Schlüsselwort *volatile* unterbindet alle Nebenläufigkeitsprobleme.
- ❏ Das Schlüsselwort *volatile* hat nur noch eine historische Bedeutung und ist in heutigen Programmen nicht mehr nötig.
- ❏ Das Schlüsselwort *volatile* erlaubt dem Compiler bessere Optimierungen durchzuführen.

e) Was ist ein Stack-Frame? 2 Punkte

- ❏ Ein bei Interrupts auftretendes spezielles Nebenläufigkeitsproblem.
- ❏ Ein Interrupt, der einen Überlauf des Stack-Speichers signalisiert.
- ❏ Ein Bereich des Speichers, in dem unter anderem Übergabeparameter für Funktionen abgelegt werden.
- ❏ Ein Bereich des Speichers, in dem lokale static-Variablen von Funktionen abgelegt werden.

f) In welcher Situation kann ein lost-update-Problem auftreten? 2 Punkte

- ❏ Bei der Modifikation von lokalen Variablen in zwei Interrupt-Handlern.
- ❏ Wenn eine Interrupt-Sperre zu lange gehalten wird.
- ❏ Wenn während einer Interrupt-Bearbeitung weitere Interrupts gesperrt sind.
- ❏ Wenn sowohl in der main-Funktion als auch in einem Interrupt-Handler modifizierend auf die gleiche `uint16_t`-Variable zugegriffen wird.

g) Welche Aufgabe erfüllt der C-Präprozessor? 2 Punkte

- ❏ Er entfernt nicht sichtbare globale Variablen aus dem Programm.
- ❏ Er bindet mehrere .o-Dateien zusammen.
- ❏ Er erzeugt einen internen Zwischencode (ähnlich dem Java-Bytecode), der dann durch den C-Compiler in ausführbaren Code umgewandelt wird.
- ❏ Er führt textuelle Ersetzungen im C-Code durch, die sich durch Makrodefinitionen steuern lassen.

Aufgabe 2a: safe (30 Punkte)

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

Schreiben Sie eine Steuerung für einen AVR-Mikrocontroller in einem Safe, der mit einem 8-stelligen Binärschlüssel gesichert ist. Der Code wird während der Eingabe durch LEDs visualisiert.

Die Eingabe des Schlüssels erfolgt mit Hilfe von zwei Tastern, die entsprechend mit "0" (Taster 0) und "1" (Taster 1) beschriftet sind. Jede der 8 Stellen des Schlüssels wird durch den Benutzer nacheinander durch Drücken eines der beiden Taster eingegeben. Die 8 LEDs visualisieren den eingegebenen Schlüssel, so dass die LEDs nach der Eingabe jeder Stelle den bisher eingegebenen Binärcode anzeigen.

Nach der achten eingegebenen Stelle wird der Schlüssel mit einer gegebenen statischen Konstante SECRET verglichen. War die Eingabe erfolgreich, öffnet sich das Schloss für eine Gesamtdauer von 5 Sekunden während alle LEDs mit einer Periodendauer von einer Sekunde (1000 ms) blinken, dann schließt das Schloss wieder. Nach dieser Sequenz oder im Fall eines falschen Schlüssels startet die Eingabe des Schlüssels wieder von vorne.

Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion `void init(void)`.
- Das Programm startet mit allen LEDs ausgeschaltet, das Schloss ist geschlossen und die Eingabe des Benutzers wird erwartet. Während des Wartens auf eine Eingabe soll der Mikrocontroller jeweils in den Schlafmodus gehen.
- Implementieren Sie das Blinken der LEDs während der Öffnung des Safes 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 gewartet werden muss, um eine Millisekunde verstreichen zu lassen.

Information über die Hardware

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

Schloss: **PORTB**, Pin 5, offen bei high-Pegel, geschlossen bei low-Pegel
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRB**-Reg. auf 1

Taster: **PORTD**, "Null"-Taster (Taster 0) = Pin 2, "Eins"-Taster (Taster 1) = Pin 3
- Pin als Eingang konfigurieren: entsprechendes Bit in **DDRD**-Register auf 0
- externe Interruptquellen **INT0** und **INT1**, ISR-Vektor-Makros: **INT0_vect** und **INT1_vect**
- Aktivierung der Interruptquellen erfolgt durch Setzen des **INT0**- bzw. **INT1**-Bits im Register **GICR**.
- die Taster verbinden den Pin mit Masse, es müssen die internen Pullup-Widerstände verwendet werden (entsprechende Bits in **PORTD**-Register auf 1 setzen).
- Konfiguration der externen Interruptquellen 0 und 1 (Bits in Register **MCUCR**)

Interrupt 0		**Beschreibung**	**Interrupt 1**	
ISC01	**ISC00**		**ISC11**	**ISC10**
0	0	Interrupt bei low Pegel	0	0
0	1	Interrupt bei beliebiger Flanke	0	1
1	0	Interrupt bei fallender Flanke	1	0
1	1	Interrupt bei steigender Flanke	1	1

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>

/* Funktionsdeklarationen, globale Variablen, etc. */
```

```
/* Unterbrechungsbehandlungsfunktionen */
```

A:

```
/* Funktion main */

   /* Initialisierung und lokale Variablen */

   /* Hauptschleife */

    /* Code-Eingabe */

    /* Schloss oeffnen, naechster Durchlauf */

/* Ende main */
```

M:

```
/* Initialisierungsfunktion */
```

```
/* Wartefunktion */
```

IW:

Aufgabe 3: (16 Punkte)

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

a) Welche zwei Verfahren gibt es, um Ereignisse eines Peripheriegeräts in einem Programm zu bearbeiten? Geben Sie eine kurze Beschreibung und nennen Sie drei Unterschiede mit Vor- bzw. Nachteilen zwischen den Verfahren. (5 Punkte)

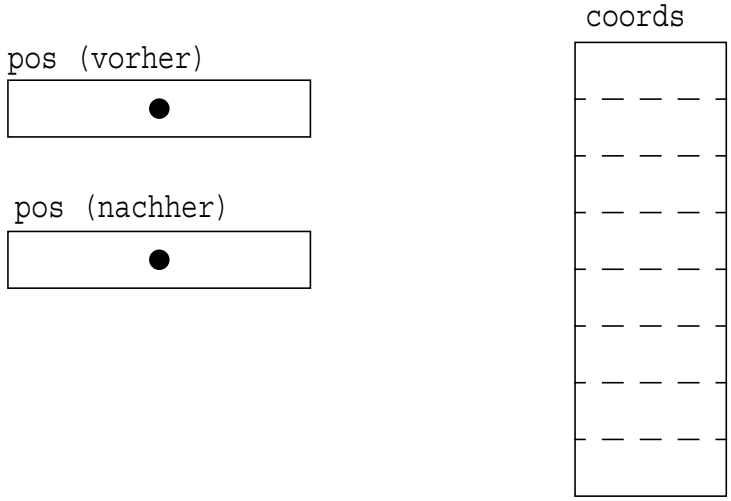
b) In Bezug auf die Semantik bei der Übergabe von Parametern an Funktionen unterscheidet man generell die Konzepte *call by value* und *call by reference.*
Beschreiben Sie die beiden Konzepte! (4 Punkte)

c) Gegeben ist folgender Code-Abschnitt:

```
struct vect {
    uint8_t x;
    uint8_t y;
};

struct vect coords[3];
struct vect *pos = &coords[1];
```

Skizzieren Sie, wie die einzelnen Elemente von `coords` im Speicher abgelegt werden. Zeichnen sie außerdem ein, an welche Speicherstelle die Variable `pos` vor und nach der Ausführung der Operation `pos++` zeigt. (4 Punkte)



d) Worin besteht der Unterschied zwischen einer globalen static-Variable und einer lokalen static-Variable in Bezug auf Nebenläufigkeit zwischen einem Programm und einem Interrupt-Handler? (3 Punkte)