Aufgabe 1: (12 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) Was versteht man unter Polling? 2 Punkte

- ❏ Ein Konzept zur Abarbeitung von Interrupts.
- ❏ Das regelmäßige Anheben eines Pegels, um einem Gerät einen bestimmten Zustand zu signalisieren.
- ❏ Wenn ein Programm zum Zugriff auf kritische Daten Interrupts sperrt.
- ❏ Wenn ein Programm regelmäßig eine Peripherie-Schnittstelle überprüft, ob Daten oder Zustandsänderungen vorliegen.

b) Welche Aussage zu Zeigern ist **richtig**? 2 Punkte

- ❏ Zeiger können verwendet werden, um in C eine call-by-reference Übergabesemantik nachzubilden.
- ❏ Die Übergabesemantik für Zeiger als Funktionsparameter ist call-by-reference.
- ❏ Ein Zeiger kann zur Manipulation von schreibgeschützten Datenbereichen verwendet werden.
- ❏ Zeiger vom Typ `void*` existieren in C nicht, da solche "Zeiger auf Nichts" keinen sinnvollen Einsatzzweck hätten.

c) Welche Aussage zu virtuellen Adressräumen ist richtig? 2 Punkte

- ❏ Die Umrechnung von virtuellen Adressen in physikalische Adressen erfolgt einmal beim Laden des Programms.
- ❏ Die Umrechnung von virtuellen Adressen in physikalischen Adressen wird zur Laufzeit durch eine spezielle Hardwareeinheit durchgeführt.
- ❏ Auf einem Rechner mit nur einer CPU müssen virtuelle Adressen nicht in physikalische Adressen umgewandelt werden, da sie nur innerhalb eines Prozesses gelten und immer nur ein Prozess ausgeführt werden kann.
- ❏ Verwendet man einen Zeiger in C, so kann man über das Schlüsselwort `static` festlegen, ob die virtuelle oder die physikalische Adressen darin gespeichert wird.

d) Was bewirkt folgende Programmanweisung: 2 Punkte

```
#define PORT (*(unsigned char *)0x3b)
PORT ^= PORT;
```

- ❏ Alle Bits in dem Register ändern ihren Wert.
- ❏ Nur die Bits mit dem Wert 0 im Register ändern ihren Wert.
- ❏ Das Register hat nach der Operation den Wert 0.
- ❏ Das Register hat nach der Operation den Wert 255.

e) Was ist ein Stack-Frame? 2 Punkte

- ❏ Der Speicherbereich, in dem der Programmcode einer Funktion abgelegt ist.
- ❏ Ein spezieller Registersatz des Prozessors zur Bearbeitung von Funktionen.
- ❏ Ein Fehler, der bei unberechtigten Zugriffen auf den Stack-Speicher entsteht.
- ❏ Ein Bereich des Speichers, in dem u.a. lokale automatic-Variablen einer Funktion abgelegt sind.

f) Gegeben sind folgende Funktionen: 2 Punkte

```
int add(int a, int b) { return a+b; }
int mul(int a, int b) { return a*b; }
```

Was ist das Ergebnis von folgendem Ausdruck:

```
5 * mul( add(4,1), 2)
```

- ❏ 22
- ❏ 30
- ❏ 42
- ❏ 50

Aufgabe 2: (30 Punkte)

Schreiben Sie ein Lauflicht-Programm für einen AVR-Mikrokontroller mit 8 LEDs an Port A. Es soll immer *genau eine LED ausgeschaltet* sein, im Ausgangszustand die LED an Pin 0. Nach Erreichen der LED an Pin 7 verlangsamt das Programm die Geschwindigkeit des Lauflichts und beginnt wieder mit der LED an Pin 0. Nach Erreichen einer minimalen Geschwindigkeit stoppt das Programm nach Ausschalten der LED an Pin 7. Mit einem Taster kann die Geschwindigkeit während des Programmablaufs erhöht werden. Die Geschwindigkeitsänderung wird sofort nach einem Tastendruck für die nächste LED-Umschaltung wirksam. Im Detail soll sich das Programm wie folgt verhalten:

- Die Initialisierung der Hardware soll in einer eigenen Funktion `void init()` erfolgen, die zu Beginn des Programms aufgerufen wird. Es können hierbei keine Annahmen über den initialen Zustand der Hardware gemacht werden.
- Danach startet ein Lauflichtdurchlauf mit der Anfangsgeschwindigkeit.
- Zum Warten zwischen dem Umschalten zweier LEDs wird die Funktion `void wait(unsigned int interval)` aufgerufen. In dieser Funktion wird aktiv in einer Warteschleife mit `interval` Durchläufen gewartet.
- Die Zahl der Schleifendurchläufe der `wait`-Funktion soll sich im Bereich [1000; 10000] bewegen und sich in 1000er Schritten nach einem Lauflichtdurchlauf bzw. einem Tastendruck verändern. In der Anfangsgeschwindigkeit ist die Zahl der Schleifendurchläufe 5000.
- Nach einem Lauflichtdurchlauf wird der Mikrokontroller in den Standard-Stromsparmodus versetzt falls die Durchlaufgeschwindigkeit das Maximum erreicht hat. Durch einen Tastendruck soll der Mikrokontroller aus dem Stromsparmodus geweckt werden können und mit der Ausgangsgeschwindigkeit beginnen.

Information über die Hardware

LEDs: **PORTA**, Pins 0-7, Start bei LED an Pin 0
- LED ist eingeschaltet, wenn das entspr. Bit im **PORTA**-Reg. auf 1 gesetzt ist
- Pin als Ausgang konfigurieren: entspr. Bit in **DDRA**-Reg. auf 1

Taster: **PORTD**, Pin 2
- Pin als Eingang konfigurieren: entspr. Bit in **DDRD**-Reg. auf 0
- externe Interruptquelle **INT0**, ISR-Vektor-Makro: **INT0_vect**
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT0**-Bits im Register **GICR**.
- Taster verbindet den Pin mit Masse, es muss der interne Pullup-Widerstand verwendet werden (entspr. Bit in **PORTD**-Reg. 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>

/* Funktiondeklarationen, globale Variablen, etc. */

.....
.....
.....
.....
.....
.....
.....
/* Unterbrechungsbehandlungsfunktion */

.....
.....
.....
.....
.....
.....



/* Funktion main */

.....
.....
  /* Initialisierung */

.....
.....
.....
.....
```

```
  /* Hauptschleife */
```

```
    /* ein Lauflicht-Durchlauf */
```

```
    /* Vorbereitung des naechsten Durchlaufs bzw. Schlafen */
```

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

L:


```
/* Funktion init */
```

```
/* Ende der Funktion init */
/* Funktion wait() */
```

```
/* Ende der Funktion wait() */
```

I:

Aufgabe 3: (18 Punkte)

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

a) Was versteht man unter einem Interrupt?

b) Wie wird ein Interrupt typischerweise bearbeitet?

c) Beschreiben Sie, wie aus einem C-Quellprogramm in 4 Schritten eine ausführbare Datei erzeugt wird (jeweils Begriff für den einzelnen Schritt angeben und eine kurze Erklärung, was in diesem Schritt passiert).

d) In welche Bereiche ist der Adressraum eines Prozesses untergliedert? Welche Arten von Variablen (Speicherklasse) werden in welchen Bereichen gespeichert?