Aufgabe 1: (10 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 ist der Unterschied zwischen den wie folgt in einer Funktion `foo` (lokal) definierten Variablen? 3 Punkte

```
void foo() {
    int a;
    static int b;
```

- ❏ Die Variable `a` ist nur für Funktionen in der Datei zugreifbar, in der `foo` definiert ist, während auf `b` auch von Funktionen in anderen Modulen des Programms zugegriffen werden kann.
- ❏ Der Speicherplatz der Variablen `a` wird jeweils beim Aufruf der Funktion `foo` 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 zugreifbar, in der `foo` definiert ist. Funktionen in anderen Modulen des Programms können prinzipiell auf `a` zugreifen, hierzu muss `a` aber in dem entsprechenden Modul mit einer `extern`-Deklaration bekannt gemacht werden.
- ❏ Da der Speicherplatz der Variablen `b` für die gesamte Ausführungszeit des Programms reserviert ist, darf es in anderen Funktionen keine weitere Variable mit dem Namen `b` geben. Bei `a` gilt diese Einschränkung nicht.

b) 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.

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

- ❏ Der Präprozessor ist eine Softwarekomponente, welche Java-Klassen durch C-Funktionen ersetzt, die dann von einem C-Compiler übersetzt werden.
- ❏ Der Präprozessor optimiert Makros durch Zeigerarithmetik.
- ❏ Nach dem Übersetzen und dem Binden müssen C-Programme durch den Präprozessor nachbearbeitet werden, um Makros aufzulösen.
- ❏ Die Syntax von Präprozessoranweisungen ist unabhängig vom Rest der Sprache C.

d) Was passiert, wenn das folgende Programmstück übersetzt und ausgeführt wird: 3 Punkte

```
unsigned char *p = (unsigned char *)0x3B;
*p = 0x01;
```

- ❏ Der Compiler wird beim Übersetzen einen Fehler melden, weil diese Art von Zugriff auf einen Zeiger nicht erlaubt ist.
- ❏ Der Datentyp `unsigned char` existiert nicht, deshalb wird der Compiler Fehler melden.
- ❏ Bei einem AVR-Mikrocontroller würde ein Bit in einem I/O-Register (in diesem Fall dem Register von Port A) gesetzt, unter Linux würde das gleiche Programm im Allgemeinen mit "Segmentation fault" abgebrochen werden.
- ❏ Unter Linux hätte das Programm keine Auswirkung, weil die Adresse 0x3B sicher nicht zu einem schreibbaren Datenbereich des Prozesses gehört. Bei einem AVR-Mikrocontroller würde der Zugriff auf die Adresse unmittelbar einen Interrupt auslösen.

Aufgabe 2a: Snake (22 Punkte)

Schreiben Sie ein Programm *Snake* für einen AVR-Mikrokontroller, welches entlang der Umrandung einer 7-Segment-Anzeige (insgesamt 6 LEDs bzw. Segmente) eine mit jedem Durchlauf um ein Segment wachsende Schlange anzeigt. Die Schlange hat initial eine Länge von einem Segment. Nach jedem Durchlauf wächst die Länge der Schlange um ein Segment. Umrandet die Schlange die komplette 7-Segment-Anzeige (Länge 6 Segmente), so stoppt die Ausführung bis zum Druck des Tasters. Ein Druck des Tasters während der Ausführung verringert die Länge der Schlange mit Wirkung zur nächsten Schlangenbewegung um ein Segment, wobei die Schlange niemals kürzer als ein Segment wird.
Das Programm soll im Einzelnen wie folgt funktionieren:

- Die main-Funktion ruft zunächst die Funktion `void init();` auf, welche die Initialisierung der I/O-Ports und Interruptquellen durchführt. Dann wird in einer Endlosschleife jeweils ein Umlauf der 7-Segment-Anzeige durchgeführt. Die tatsächliche Anzeige der Schlange wird in der **vorgegebenen** Funktion `void showSnake(int pos, unsigned char len);` durchgeführt, welche als Parameter die Position des Schlangenkopfes (Wert im Bereich [0;5]) und die Länge der anzuzeigenden Schlange erhält (Wert im Bereich [1;6]).
- Zwischen einer Bewegung der Schlange wird jeweils eine Wartezeit durch eine aktive Warteschleife mit 5000 Durchläufen realisiert. Die Warteschleife soll in eine Funktion `void active_wait(volatile unsigned int len);` ausgelagert werden, deren Parameter die Anzahl der Schleifenläufe angibt. Nach einer Umrundung der Anzeige erfolgt keine gesonderte Wartezeit.
- Ist die maximale Länge der Schlange erreicht, so wird der Mikrokontroller in den Standard-Stromsparmodus versetzt und wartet passiv auf einen Druck des Tasters. Nach Drücken des Tasters startet die Ausführung erneut mit Schlangenlänge 1.
  Beispielgrafik: Schlange mit Kopfposition 0, Schlangenlänge 4

Information über die Hardware

LEDs: **PORTB**, Pins 0-5, Start bei LED an Pin 0, aktiviert bei low-Pegel
- Pin als Ausgang konfigurieren: entspr. Bit in **DDRB**-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>

#define NUMLED 6

/* Funktiondeklarationen, globale Variablen, etc. */

-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------------------------
/* Unterbrechungsbehandlungsfunktion */

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

/* Funktion main */

-----------------------------------------------------------------------
-----------------------------------------------------------------------
  /* Initialisierung */

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

```
  /* Hauptschleife */
```

```
    /* ein Snake-Umlauf */
```

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

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

L:


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

```
/* Ende der Funktion init */

/* Funktion active_wait() */
```

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

I:
W:

Aufgabe 3: (13 Punkte)

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

a) Beschreiben Sie den Ablauf einer Interrupt-Bearbeitung wie sie z. B. auf einem AVR-Mikrocontroller oder einem ähnlichen Prozessor erfolgt.

b) Beschreiben Sie das *Lost-Update*-Problem anhand eines konkreten Beispiels!