# Woche 09: Hausaufgabe 05

## Allgemeines

Die zu erstellenden Funktionen sollen in der Datei `task.c` definiert werden. Innerhalb der Datei existieren bereits die Funktionsköpfe und Rümpfe (leer - sollen von euch gefüllt werden). Hierbei ist zu beachten:

- Die Funktionsköpfe dürfen **NICHT** verändert werden. Dies gilt auch für die Argumente.
- Die zugehörige Headerdatei `task.h` ist mit den entsprechenden Funktionsdeklarationen zu füllen. Zusatzfunktionen dürfen nicht enthalten sein.
- Innerhalb der `task.c` ist es verboten, eine `main`-Funktion zu erstellen. Testet die Funktionen innerhalb der gegebenen Datei `test.c`. Innerhalb der Datei `test.c` könnt ihr machen was ihr möchtet.
- Es dürfen **KEINE** globalen Variablen verwendet werden.
- Abzugeben sind nur die Datei `task.h` und `task.c`. Wir rufen die Funktionen in unserer eigenen `main`-Funktion auf. Es werden auch nur diese Dateien bewertet.
- Solange nicht **EXPLIZIT** gefordert, sind die Nutzung von Userausgaben (printf etc.) und Usereingaben (scanf etc.) innerhalb der zu erstellenden Funktionen untersagt.
- Der Code wird von uns mit `gcc -Wall -Wextra <sourcedateien>` kompiliert. Dieser zeigt euch Warnungen an, d.h. der Compiler wird sich beschweren, jedoch eine ausführbare Datei, sofern keine Fehler vorhanden sind, erzeugen.
- Abgegebene .c Dateien, welche nicht kompiliert werden können, werden mit 0 Punkten bewertet.
- Alle Funktionsköpfe , auch wenn sie nicht gelöst werden, `müssen` in der Header Datei enthalten sein, d.h. alle Funktion müssen in einer möglichen `main` aufrufbar sein.
- Eine Funktion, welche einen Rückgabewert besitzt (`void` zählt nicht dazu), **muss** ein `return` Statement mit einem passenden Wert besitzen.

## Aufgabe (1) Linearer Kongruenzgenerator — 1 Punkt

Programmieren Sie einen linearen Kongruenzgenerator. Für die Konstanten `MULTIPLIER`, `OFFSET`, `SEED` und `MODULOCONST` sollen in der `task.h` vier *symbolische Konstanten* angelegt werden. Die Berechnungsvorschrift lautet:

$$x_{k+1} = (\texttt{MULTIPLIER} \cdot x_k + \texttt{OFFSET}) \; \textit{modulo} \; \texttt{MODULOCONST}$$

für $k \in \mathbb{N}$

Dieser wird dazu verwendet, Pseudozufallszahlen zu generieren, ausgehend von einem Startwert $x_0$, welcher auch als *seed* bezeichnet wird.

Nutzen Sie hierfür den Funktionskopf der `task.c`:
`unsigned long lcg(void)`

Die symbolischen Konstanten `MULTIPLIER`, `OFFSET`, `SEED` und `MODULOCONST` können frei gewählt werden.

Ein Beispielaufruf könnte wie folgt aussehen:

```
1  // MULTIPLIER -> 397204094
2  // OFFSET -> 0
3  // MODULOCONST -> 2147483647
4  // SEED aka x0 -> 58854338
5
6  // Erster Aufruf -> Rueckgabe von x1
7  printf("%lu\n",lcg());
8  // 1292048469 <- x1
```

## Aufgabe (2) Richtungslogik — 1 Punkt

In dieser Aufgabe soll ausgehend von zwei Winkeln, dem aktuellen Winkel `is` und dem Sollwinkel `target`, die kleinste Winkeldifferenz, inklusive Vorzeichen zurückgegeben werden. Das Vorzeichen kennzeichnet hierbei die Drehrichtung. Die Winkel `is` und `target` liegen im Bereich $[0, 360[$. Es sollen nur ganze Zahlen betrachtet werden. Die Funktion soll den nachfolgenden $\Delta$-Wert zurückgeben.

Nutzen Sie hierfür den Funktionskopf der `task.c`:
`int directionlogic(int is, int target)`

Drei Beispiele:

1) $is = 50° \quad target = 120° \Rightarrow \Delta = 70°$
2) $is = 10° \quad target = 350° \Rightarrow \Delta = -20°$
3) $is = 180° \quad target = 30° \Rightarrow \Delta = -150°$

Zwei weitere Beispiele in Codeform:

```
1   ...
2   int delta = directionlogic(350,348);
3   // delta -> -2
4   delta = directionlogic(10,25);
5   // delta -> 15
```

***Hinweis:*** *Die Nutzung von Funktionen aus der* `<stdlib.h>` *sind erlaubt.*

## Aufgabe (3) Datei einlesen — 1 Punkt

In dieser Aufgabe soll eine beliebige Textdatei eingelesen und in ein Charakterarray mit maximal 5000 Zeichen gespeichert werden. Gleichzeitig sollen alle eingelesenen Zeichen gezählt und die Gesamtanzahl als Rückgabewert zurückgegeben werden.

Nutzen Sie hierfür den Funktionskopf der `task.c`:
`int readFile(char filename[], char textarray[5000])`

Beispieldatei: Text.txt

```
Hallo
Studierende im Kurs EIT
```

Beispielaufruf:

```
1   char textarray[5000] = {0};
2   int count = 0;
3   count = readFile("Text.txt", textarray);
4   // count -> 31
5   printf("%s",textarray); // -> identisch zu Text.txt
```

***Hinweis:*** *Ein Zeilenumbruch enthält zwei Zeichen. Beachten Sie, dass das String-Ende auch ein Zeichen darstellt. Nutzen sie die Funktion* `fgetc`

## Aufgabe (4) cReEpYtExT — 1 Punkt

In dieser Aufgabe soll eine beliebige Zeichenkette in einen *cReEpY*-Text umgewandelt werden. Hierzu soll die Zeichenkette durchlaufen werden. An jedem geraden Index der Zeichenkette soll, sofern es sich um einen Buchstaben handelt, ein kleiner Buchstabe stehen, an jedem ungeraden Index ein großer Buchstabe (gehen Sie davon aus, dass im Text keine Umlaute vorkommen).

Nutzen Sie hierfür den Funktionskopf der `task.c`:
`void cReEpYtExT(char text[])`

Beispielaufruf:

```
1  char text[] = "This is black magic!11";
2  cReEpYtExT(text)
3  printf("%s",text); // -> tHiS Is bLaCk mAgIc!11
```

***Hinweis:*** *Sie können für diese Aufgabe Funktionen der* `<ctype.h>` *verwenden.*

## Aufgabe (5) Datum und Tageszeit (Zusatz) — 1 Punkt

In dieser Aufgabe, sollen sie eine Funktion programmieren, welche die aktuelle Tageszeit im Terminal ausgibt. Für DIESE Aufgabe sollen Sie `printf` innerhalb dieser Funktion benutzen! Die Ausgabe soll folgendes Format haben (Führende Nullen sind mit anzugeben):

`YYYY/MM/DD - HH:MM:SS Uhr` Beispiel: `2021/01/11 - 09:45:38 Uhr`

Nutzen Sie hierfür den Funktionskopf der `task.c`:
`void printTime(void)`

Ein Beispielaufruf kann wie folgt aussehen:

```
1        ...
2        printTime();
```

Ausgabe im Terminal bei Aufruf der Funktion:

```
2021/01/11 - 15:30:23 Uhr
```

***Hinweis:*** *Sie benötigen für diese Aufgabe die* `<time.h>`*. Nutzen Sie das Internet um diese Aufgabe zu lösen.*