www.thomasgalliker.ch

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Mikrocontroller, C-Programmierung

Formativer Test

P. Sollberger; Version 2.1

Name: Galliker Vorname: Thomas

Unterschrift: _____

Rahmenbedingungen:

1. Prüfungszeit: **30 Minuten**
2. In der Prüfung können maximal **30 Punkte** erreicht werden. Jeder Aufgabe ist eine maximal erreichbare Punktezahl zugeordnet.
3. Schreiben Sie Ihren Namen auf dieses Blatt und unterschreiben Sie dieses Blatt. Blätter ohne Namensangabe und Unterschrift werden nicht bewertet. Mit Ihrer Unterschrift bestätigen Sie, dass Sie die Prüfung persönlich und unter Einhaltung aller Reglemente ausgeführt haben.
4. Es handelt sich um eine schriftliche Prüfung ohne Einsatz des Computers oder elektronischer Hilfsmittel. Sie dürfen keine Unterlagen verwenden.
5. Sollte die Problemstellung Unklarheiten aufweisen, dürfen Sie eigene Annahmen treffen. Führen Sie diese in der Lösung auf.
6. Schreiben Sie möglichst verständlich und gut leserlich. Missverständliche Lösungen werden nicht berücksichtigt.
7. Benutzen Sie den Freiraum unter den Aufgaben für Ihre Lösung.

Für die Korrektur (nicht ausfüllen!)

1	2	3	4	5	Punkte

Aufgabe 1: Ausdrücke (4 Punkte)

Bestimmen Sie die Ausgabe folgender Zeilen (1 Punkt pro Zeile → 4 Punkte).

```
int a = 12;
unsigned int b = 11;

printf("%d\n", a++);
```

12

```
printf("%d\n", b == 2345);
```

0 = false
1 = true

0

```
printf("%u\n", b << 2);
```

00001011 → 2x shift left 00101100

Achtung: Funktioniert nur mit unsigned int, wegen Vorzeichen-Bit

b<<x ⇒ b*2^x

44

```
printf("%s\n", ((a < 20) ^ (b > 0)) ? "true" : "false");
```

(a < 20) = true; ^ XOR; (b > 0) = true

true

Aufgabe 2: Typendefinition (2 Punkte)

Definieren Sie Ihren eigenen Boolean Typ, welcher aus der Aufzählung (Enumeration) der beiden Werte FALSE (Wert 0) und TRUE (Wert 1) besteht.
Der neue Typ soll anschliessend folgendermassen angewendet werden können:

```
Boolean_t b = TRUE;
```

(2 Punkte).

obsolete Bezeichnung

```
typedef enum Boolean {
    FALSE = 0,
    TRUE = 1
} Boolean_t;
```

AND = &&
bit-weise AND = &

OR = ||
bit-weise OR = |

XOR = ^

Boolean_t var1 = ...
aber
enum Boolean var2 = ...

Aufgabe 3: call-by-reference Parameter (7 Punkte)

1. Schreiben Sie eine Funktion `swap`, mit der Sie zwei int-Werte vertauschen können. Die Werte sind dauerhaft zu vertauschen. Die Funktion gibt keinen Wert zurück, ist also vom Typ `void`.
(5 Punkte)

```
void swap (int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

2. Gegeben sind folgende Variablendefinitionen:

```
int wert1 = 14;
int wert2 = 13;
```

Rufen Sie Ihre Funktion für diese beiden Variablen auf:
(2 Punkte)

```
swap(&wert1, &wert2);
```

=> Adresse muss übergeben werden
=> Bei Vektoren ist das &-Zeichen überflüssig

Aufgabe 4: Header- und Sourcecode-Dateien (7 Punkte)

Gegeben ist der Source Code auf der folgenden Seite.
Dieser Code soll auf drei Dateien verteilt werden: eine Header und zwei Sourcecode Dateien.

Die Dateien beinhalten folgende Elemente:

main.h	- Definition der enum Werte sowie der globalen Variable - Funktionsdeklarationen
main.c	- Funktion main
func.c	- Funktionsimplementationen

Aufgaben:

1. Markieren Sie im Source Code auf der folgenden Seite deutlich den Code, der in die Datei func.c verschoben werden soll.
   Nehmen Sie notwendigen Ergänzungen des Codes vor.
   (2 Punkte)

2. Markieren Sie im Source Code auf der folgenden Seite deutlich den Code, der in die Datei main.c verschoben werden soll.
   Nehmen Sie notwendigen Ergänzungen des Codes vor.
   (1 Punkt)

3. Schreiben Sie untenstehend die **vollständige** Datei main.h hin.
   Vermeiden Sie, dass diese Datei mehrfach „includiert" werden kann.
   (4 Punkte)

```
#ifndef FUNC_H  #define FUNC_H
int eingabe(void);
int quadrat(int);          //besser wäre int quadrat(int zahl);
extern int errorCode;
enum error {ERROR, OK};
#
```

```
#include<stdio.h>

enum error {ERROR, OK};

int errorCode=OK;          //kann in func.c und main.c sein
#include "main.h"
int eingabe(void)
{
    int zahl;

    printf("Gib Zahl ein: ");
    scanf("%d", &zahl);

    if (zahl <= 0)
    {
        errorCode = ERROR;
    }

    return zahl;
}


int quadrat(int zahl)
{
    return zahl * zahl;
}
#include <stdio.h>
#include "main.h"          //sollte besser "func.h" heissen
int main (void)
{
    int wert;

    wert = eingabe();

    if (errorCode == ERROR)
    {
        printf("Die Eingabe war falsch!\n");
    }
    else
    {
        printf("%d im Quadrat ist %d\n", wert, quadrat(wert));
    }
    return 0;
}
```

Aufgabe 5: Pointer auf Array (10 Punkte)

Schreiben Sie die Methode fillArray, welche einen Vektor von `numValues` int-Werten alloziert und ihn mit den Werten `0... numValues-1` füllt. Die Methode liefert `SUCCESS` zurück, falls die Speicherallozierung erfolgreich war, ansonsten `FAILURE`.
Im Code unten sehen Sie, wie die Methode angewendet wird und die dazugehörende Ausgabe.

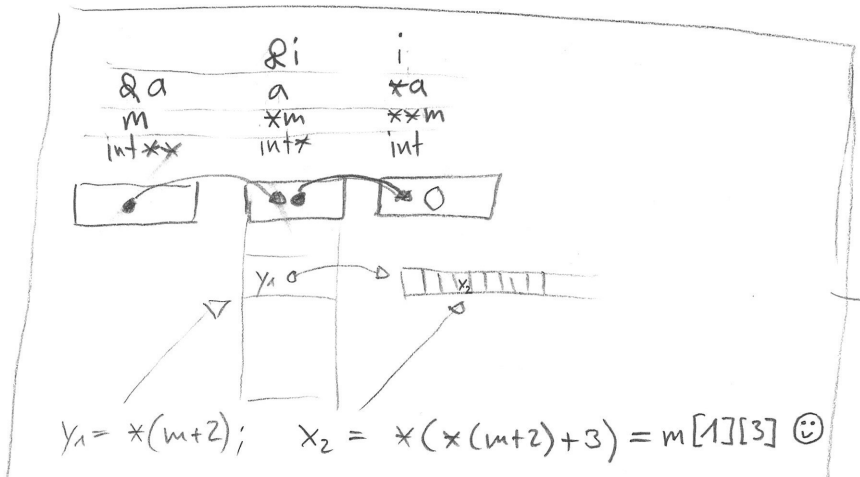
```
#define NUM_VALUES 3
enum FailureCode {SUCCESS, FAILURE};
main()
{
    int i, *a;

    if (fillArray(NUM_VALUES, &a) == SUCCESS)
    {
        for (i = 0; i < NUM_VALUES; i++)
            printf("%dter Wert: %d\n", i, *(a+i));
        free(a);
    }
}
```

Konsolenausgabe:

```
0ter Wert: 0
1ter Wert: 1
2ter Wert: 2
```

```
enum FailureCode fillArray(int numValues, int** a)
{
  int *p;
  int i;
*p = (int*) malloc (sizeof(int) * numValues);
  if(*p) //ungleich 0
  {
    for(i=0; i < numValues; i++)
    {
      *(*a+i)=i;
    }
    return SUCCESS;
  }
  else {
    return FAILURE;
  }
}
```



$y_1 = *(m+2);$ $x_2 = *(*(m+2)+3) = m[1][3]$ ☺

Identische Schreibweise

```
x = *(a+2);
x = a[2];
```