

Klausurdeckblatt



Matrikel – Nr.:

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

Bitte tragen Sie Ihre Matrikelnummer und Ihren Namen in die dafür vorgesehenen Felder ein. Bitte in deutlicher Handschrift mit einem schwarzen Stift (nicht Bleistift). Das Feld mit dem **Barcode ist unbedingt frei zu lassen.**

Vorname:

Nachname:

Danke.

Der Bereich unterhalb dieser Linie kann von der Fakultät frei gestaltet werden.

Einführung in die Programmierung

16. Dezember 2025

M O C K – E X A M

Studiengang:

☐ Physik

☐ ET/IT

☐ NES

☐ -----

Hinweise:

Sie haben zur Bearbeitung dieser Probeklausur 90 Minuten zur Verfügung. Hilfsmittel jeglicher Art (mit Ausnahme dokumentenechter Schreibgeräte) sind nicht erlaubt. Die Aufsicht gibt Ihnen keine Hilfestellung bei der Lösung dieser Aufgaben und wird entsprechende Anfragen von Ihrer Seite nicht beantworten (mit Ausnahme von sprachlichen Verständnisfragen).

Für das „Bestehen“ dieser Probeklausur sind mindestens **15 Punkte** erforderlich.

Aufgabe	1	2	3	4	Σ
Erzielte Punkte					
Erreichbare Punkte	6	4	10	10	30

Nutzen Sie die Rückseiten der Blätter ggf. für Rechenwege. Sie sollten sich zuerst alle Aufgaben durchlesen und dabei auf die von Ihnen geforderten Ergebnisse achten, bevor Sie versuchen, die Aufgaben zu lösen. Tragen Sie Ihre Antworten unbedingt in die dafür vorgesehenen Felder ein. Nur Eintragungen innerhalb dieser Felder werden gewertet. **Viel Erfolg!**

Aufgabe 1: Darstellung von Daten [6 Punkte]

a) Geben Sie die Zahl 77 als 8-Bit breite, vorzeichenlose Binärzahl an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das höchstwertigste Bit (MSB) und das rechteste das niederwertigste Bit (LSB) der Zahl enthalten soll:

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

b) Geben Sie die Zahl -77 als 8-Bit breite Binärzahl in der Zweierkomplementendarstellung an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das Vorzeichenbit und das rechteste das niederwertigste Bit der Zahl enthalten soll:

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

vz

c) Gegeben seien die beiden 8-Bit breiten Binärzahl 0b01101100 und 0b00001001. Beide Zahlen sind in Zweierkomplementendarstellung dargestellt. Berechnen Sie die Summe beider Zahlen und geben Sie diese als 8-Bit breite Binärzahl in Zweierkomplementendarstellung an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das Vorzeichenbit und das rechteste das niederwertigste Bit der Summe enthalten soll:

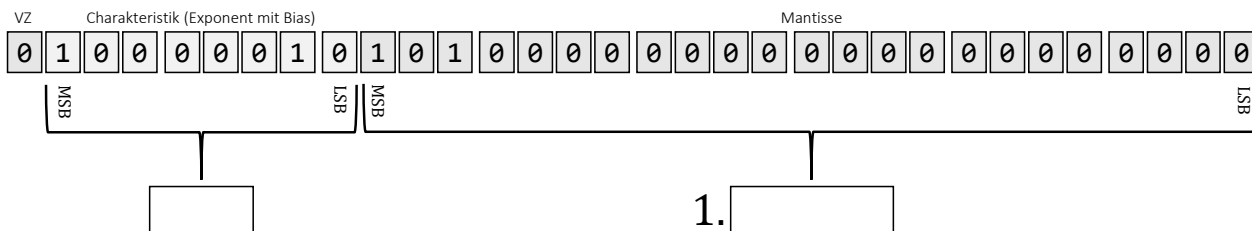
--	--	--	--	--	--	--	--

d) Berechnen Sie den Quotienten der beiden 8-Bit breiten Binärzahlen aus der Teilaufgabe c) und geben Sie diesen wieder im selben Format als Binärzahl an. Nutzen Sie die größere der beiden Zahlen als Dividend und die kleinere als Divisor. Bei korrekt durchgeführter Division sollten Sie in dieser Aufgabe den Rest 0 erhalten:

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

Aufgabe 2: Fließkommazahlen [4 Punkte]

a) Betrachten Sie das nachfolgende Bitmuster einer IEEE 754 Fließkommazahl (Einfache Genauigkeit, beispielsweise float) und bestimmen Sie den Dezimalwerte der Mantisse sowie den Dezimalwert des Exponenten mit Bias (127). Stellen Sie anschließend den Wert der dargestellten Fließkommazahl sowohl in Exponentendarstellung als auch in Form einer einfachen Dezimalzahl dar. Tragen Sie die Ergebnisse in die dazu vorgesehenen Felder ein.



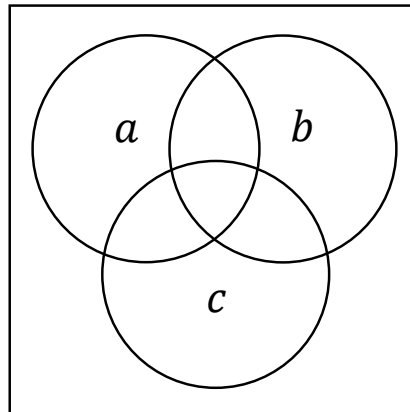
$$= \circ 2^{\square} \cdot 1. \square = \square$$

b) Was bedeutet NaN im Kontext des Ihnen bekannten Gleitkommazahl-Standards (IEEE32) und warum ist solch eine Darstellung sinnvoll?

Aufgabe 3: Boolesche Logik und Bitoperationen [10 Punkte]

a) Stellen Sie den folgenden Term von Bitoperationen im Mengen-Diagramm dar:

$$\sim(a \wedge b \wedge c)$$



b) Schauen Sie sich den nachfolgenden Codeausschnitt an. Das untenstehende Raster repräsentiert das Array `matrix`. Malen Sie alle Zellen des Rasters aus, wenn sich an dieser Stelle im Array nach der Ausführung des Programmabschnitts eine 1 befindet:

```
bool matrix[8][7];

for (int i = 0; i < 8; ++i) {
    matrix[i][3] = i % 2;
    matrix[i][2] = (i >> 1) % 2;
    matrix[i][1] = i >> 2;
    matrix[i][0] = matrix[i][2] && matrix[i][1];
    matrix[i][2] = matrix[i][2] || matrix[i][1];
    for (int j = 0; j < 4; ++j) { matrix[i][6-j] = matrix[i][j]; }
}
```

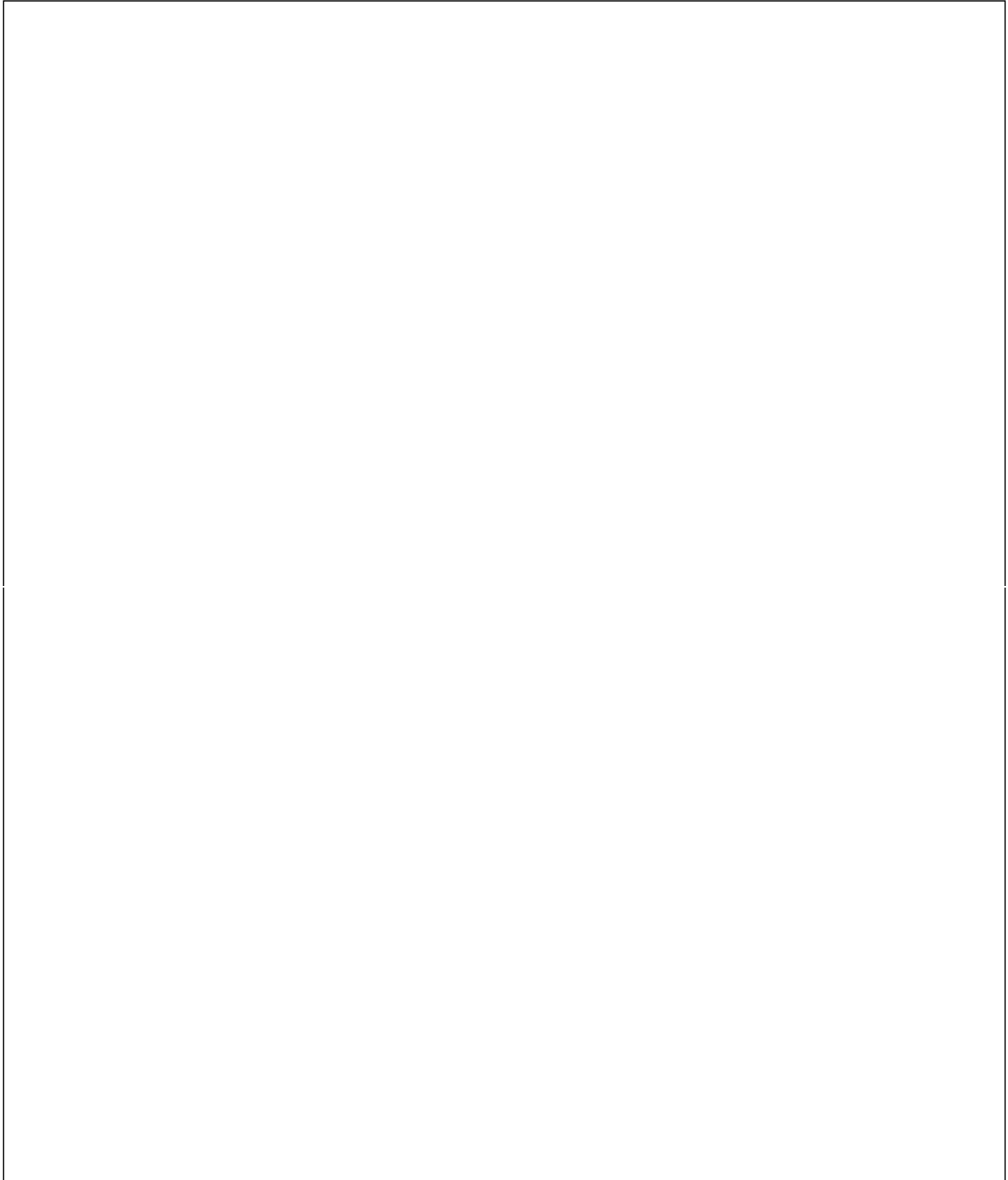
matrix[i][j]		j	0	1	2	3	4	5	6
i	0								
	1								
	2								
	3								
	4								
	5								
	6								
	7								

Name:

Matrikelnummer:

c) Implementieren Sie eine Funktion `void f(uint16_t* x, uint8_t n)`, welche das n -te Bit des Bitmusters von `*x` toggelt. Falls $n \geq 16$ gilt, soll `f` einfach nichts tun.

```
void f(uint16_t* x, uint8_t n) {
```



```
}
```

Aufgabe 4: Speicher und Zeigerarithmetik [10 Punkte]

a) Im folgenden Hauptprogramm werden die Werte von Variablen mit Hilfe von Zeigern verändert. Nehmen Sie an, dass ein `int`-Wert genau 4 Bytes groß ist.

```
int main() {
    // Die Adresse von a[0] ist 1000
    int a[] = { 40, 60, 100 };
    int b = 10;      // &b ist 2000
    int *p = &b;
    int *q = a;
    //1

    *q = *p;
    //2

    q = p;
    p++;
    p = &a[1];
    *p = 20;
    //3

    p += 1;
    *q = *p;
    //4

    *p = *(p-1)* *q;
    //5

    return 0;
}
```

Hinweis: Die Werte der Variablen nach Schritt 1 sind bereits vorgegeben. Dadurch sind insbesondere auch die *Adressen* der Variablen gegeben.

Wir interessieren uns für die Werte der Variablen zum Zeitpunkt der mit `//1`, `//2` usw. gekennzeichneten Zeilen. Vervollständigen Sie bitte die folgende Tabelle:

Schritt	a[0]	a[1]	a[2]	p	*p	q	*q
1	40	60	100	2000	10	1000	40
2							
3							
4							
5							

Name:

Matrikelnummer:

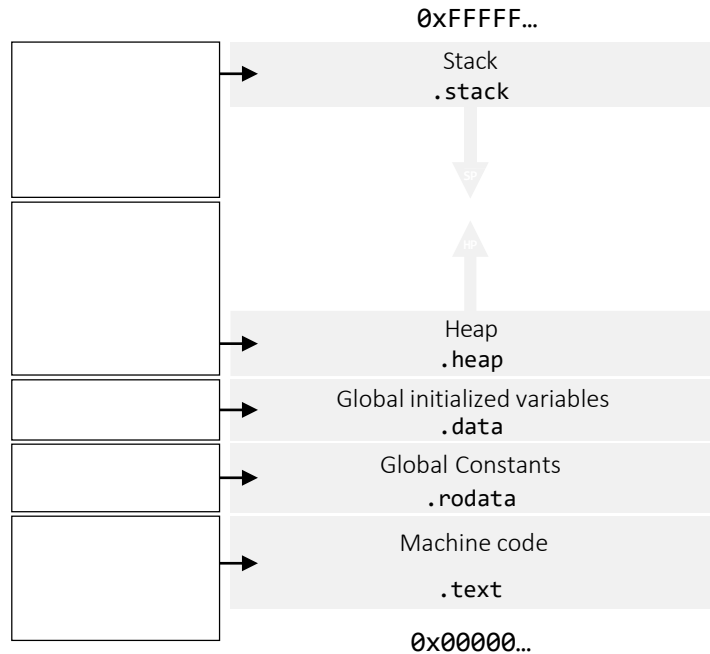
b) Schauen Sie sich das folgende C-Programm an:

```
#include <stdio.h>

float a = 5.0;
float b = 3.0;
const float c = 42;

float m(float x, float y) {
    float z = (x + y)/2.0;
    return z;
}

int main() {
    printf("%f;", m(a,m(b,c)));
    printf("\n%f\n\r", c);
    return 0;
}
```



Markieren Sie alle Bezeichner in dem oben gegebenen C-Code und Ordnen Sie diese dem jeweils passenden Speichersegment im gegebenen Speicherlayout zu, indem Sie die entsprechenden Bezeichner in die Antwortfelder eintragen.

c) Im folgenden C-Programmcode wird ein dreidimensionales Array deklariert. Ein `uint16_t`-Wert ist 2 Bytes groß. Angenommen das Element `block[0][0][0]` besitzt die Adresse 53. Welche Ausgabe erzeugt das Programm?

```
#include <stdio.h>
#include <stdint.h>

uint16_t block[3][4][5];

int main() {
    printf("%03d", &block[2][1][4] - &block[0][0][0]);
    return 0;
}
```

Schreiben Sie die Konsolenausgabe des dargestellten Programms in das nachfolgende Kästchen:

Berechnen Sie die Adresse des Elements `block[2][1][4]` und schreiben Sie diese als dreistellige Dezimalzahl in das nachfolgende Kästchen: