Name: **Vorname:** **Matr.-Nr.:**

Hochschule Kaiserslautern
FB Informatik und Mikrosystemtechnik
Prof. Dr. M. **Duque-Antón**

Bachelor-Klausur im SoSe 2023

im Fach

Grundlagen der Informatik

Angewandte Informatik / Medieninformatik / Medizininformatik

Datum: **17. Mai 2023**

Es sind **keinerlei Hilfsmittel** zur Klausur zugelassen.

Achtung: Bitte schreiben Sie die **Lösung unter die Aufgabe in das Aufgabenblatt**. Wenn Sie mehr Platz benötigen, legen Sie **von der Aufsicht** zu erhaltende **zusätzliche Blätter** dazwischen.

Die schriftliche Prüfung besteht aus 5 Aufgaben. Schreiben Sie bitte auf **jedes Blatt Ihren Namen und Matrikelnummer**. Es werden nur Blätter mit Namen und Matrikelnummer korrigiert oder bewertet. Unleserliche Lösungen, Lösungen mit Bleistift oder Rotstift werden nicht korrigiert oder bewertet.

1	2	3	4	5	Summe	Note[a]
/23	/25	/23	/29	/20	/120	

a.Eine 1.0 gibt es ab 100 Punkte, eine 5.0 unter 50 Punkte.

1. Aufgabe: Allgemeine Grundlagen

Gegenstand dieser Aufgabe sind allgemeine Themen aus den Bereichen Rechnerarchitektur, Betriebssysteme, Programmierung und Design. Konkret sollen die folgenden Fragen kurz beantwortet werden:

(a) Erläutern Sie das **Single Responsibility-Prinzip**. Erläutern Sie auch den entsprechenden Vorteil!

(b) Welchen Wert hat der Ausdruck (n++ + m) und welche Werte haben die Variablen nach der Auswertung, wenn n den Anfangswert 2 und m den Anfangswert 6 hat?

(c) Bitte stellen Sie eine beliebige for-Schleife mit Hilfe einer while-Schleife dar. Verwenden Sie dazu folgende allgemeine Darstellung:

```
for (A; B; C) {
    D;
}
```

(d) Welche **Daten-Typen** werden in Java im **Zweierkomplement** dargestellt? Wie sieht der entsprechende **Zahlenbereich** aus, der mit Hilfe von n Bit dargestellt werden kann?

(e) Erläutern Sie die beiden Cast-Varianten, implizit und explizit, an Hand von jeweils einem kleinen Beispiel (primitive Datentypen)!

(f) In der Programmiersprache Java existiert unter anderen das Schlüsselwort **this**. In welchen beiden semantischen Zusammenhängen wird dieses Schlüsselwort eingesetzt?

(g) Erläutern Sie die Funktionsweise des **Heap**, nehmen Sie dabei insbesondere Bezug auf die Besonderheit in der Programmiersprache Java!

1.)	a	b	c	d	e	f	g	Summe
Punkte	/3	/3	/4	/3	/3	/3	/4	/23

2. Aufgabe: Zahlendarstellungen und Sprachen

Gegenstand dieser Aufgabe ist die **Darstellung** von **Zahlen** innerhalb eines Rechners, der Umgang mit den **Operatoren** auf der **Bit-Ebene** und die **Beschreibung von Sprachen** auf der Basis von **BNF** (Backus Naur Form) und **EBNFs (**Extended Backus Naur Form**)**.

Hinweis: In **Java** können **Literale** auf der Basis verschiedener **Basissysteme** bearbeitet werden. Mit den Präfixen **0**, **0b** und **0x** sind jeweils die Zahlensysteme **Oktal, Binär** bzw. **Hexadezimal** gemeint. Bitte beachten Sie auch, dass in Java **Ganzzahlen Literale** implizit vom Typ int sind.

(a) Gegeben seien die folgenden Java-Anweisungen. Wie sieht die entsprechende Ausgabe auf dem Bildschirm aus?

```
System.out.println("a = " + 0b00000110);

System.out.println("b = " + 00000110);

System.out.println("c = " + 0x00000110);

System.out.println("d = " + (byte) 0b10000110);
```

(b) In dieser Teilaufgabe wird die Anwendung der Operatoren auf der Bit-Ebene überprüft. Auch hier wieder die Frage, wie sieht die entsprechende Ausgabe auf dem Bildschirm aus?

```
System.out.println("e = " +  (byte) ~43);

System.out.println("f = " +  (5 & 13));

System.out.println("g = " +  (10 ^ 15));
```

(c) Betrachten Sie zunächst das Zahlensystem zur Basis 3, welches genau mit Hilfe des Alphabets T = {0, 1, 2} dargestellt werden kann. Gesucht ist die Teilmenge, welche nur solche Muster enthält, welche dezimal durch 3 teilbar sind.

Bestimmen Sie zunächst die gesuchte Sprache. L(G) soll insbesondere die Darstellung der Zahl mit dem Wert 0 enthalten, führende Nullen sind nicht erlaubt.

Geben Sie anschließend eine Grammatik G = (N, T, P, S) mit T = {0, 1, 2} an so, dass die erzeugte Sprache L(G) die Menge aller durch 3_{10} teilbaren Zahlen zur Basis 3 darstellt. Wählen Sie N und P möglichst minimal!

2.)	a	b	c	Summe
Punkte	/8	/9	/8	/25

3. Aufgabe: Java-Programmierung

Gegenstand dieser Aufgabe ist die Programmierung mit Hilfe der **Programmiersprache Java**, insbesondere die Verwendung von **Rekursion** und **Iteration**.

(a) Gegeben sei die folgende **Klasse Aufgabe3a**, welche bereits die Methode iterativ enthält. Fügen Sie bitte eine **weitere Methode** mit dem Namen **rekursiv hinzu**, welche **dieselbe Funktionalität** liefert aber nur mit **Hilfe der Rekursion** arbeitet.

```
class Aufgabe3a {

    public static void main (String [] args) {
            int eingabe = Integer.parseInt (args[0]);
            System.out.println ("Das iterative Ergebnis lautet: " + iterativ (eingabe));
            System.out.println ("Das rekursive Ergebnis lautet: " + rekursiv (eingabe));
    } // main

    static int iterativ (int n) {
            if (n < 0) return  -1;
            int result = 1;
            for (int i = 1; i <= n; i++)      result *= i;
            return result;
    } // iterativ









} // Aufgabe3a
```

(b) Ein funktionierendes Java-Programm ist durcheinander geraten. Es soll die folgende Ausgabe erzeugen

:

java Aufgabe3b
0 6
0 5
1 6
1 5
3 6
3 5

Die einzelnen Codeteile des Programms sind in folgender Liste ungeordnet angegeben.

x++
if (x == 2)
System.out.println (x + " " + y)
class Aufgabe3b
for (int y=6; y > 4; y--)
for (int x=0; x <= 3; x++)
public static void main (String[] args)

In der Unordnung sind leider auch alle geschweiften Klammern und Semikolons verloren gegangen. Bauen Sie aus den Bausteinen, Semikolons und geschweiften Klammern wieder ein korrektes Programm, das die obige Ausgabe erzeugt! Achten Sie dabei auf korrekte Einrückungen.

(c) Gegeben sei die folgende Klasse WasWohl. Was wird beim Aufruf **java WasWohl** auf dem Bildschirm ausgegeben?
Hinweis: Die **Klasse String** bietet unter anderen die Methode **char[ ] toCharArray ()** und den Konstruktor **String (char[ ] value) an.** In der **Dokumentation** steht: The method **toCharArray ( )** converts this string to a new character array, **String (char[ ] value)** allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

```
class WasWohl {

    public static String methode (String s, int n) {
        char [] ca = s.toCharArray ();
        for (int i=0; i< ca.length; i++) ca [i] += n;
        return new String (ca);
    } // methode

    public static void main (String [] args) {
        System.out.println (methode ("Dcgl", 2));
    } // main

} // WasWohl
```

3.)	a	b	c	Summe
Punkte	/7	/8	/8	/23

4. Aufgabe: Java-Programmierung

Gegenstand dieser Aufgabe ist die Programmierung mit Hilfe der Programmiersprache Java, insbesondere die Verwendung interner Datenstrukturen und der Umgang mit Ein- und Ausgabe-Methoden in Java.

(a) Im Default Directory sind aktuell die beiden Dateien **Aufgabe4a.java**, **Aufgabe4a.class** und **eingabe.txt** enthalten. Wie sieht die **Ausgabe in der Konsole** aus, wenn das entsprechende Programm aufgerufen wird, also nach dem Befehl **java Aufgabe4a**?

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Aufgabe4a {
    public static int [][] methode1 (String datei)  throws FileNotFoundException {
        Scanner eingabe = new Scanner (new File  (datei));
        int [][] matrix;
        matrix = new int [eingabe.nextInt()][];
        for (int i=0; i<matrix.length; i++) {
            matrix[i] = new int [eingabe.nextInt()];
            for (int j=0; j < matrix[i].length; j++)
                matrix[i][j] = eingabe.nextInt();
        }
        eingabe.close ();
        return matrix;
    } // methode1

    public static int [] methode2 (int [][] m)   {
        int [] v = new int [m.length];
        for (int i=0; i < v.length; i++)
            for (int j=0; j < m[i].length; j++)
                v[i] += m[i][j];
        return v;
    } // methode2

    public static void methode3 (int [] v)   {
        for (int i=0; i < v.length; i++)
            System.out.println (v[i]);
    } // methode3

    public static void main (String [] args)  throws FileNotFoundException {
        methode3 (methode2 (methode1 ("eingabe.txt")));
    } // main
} // Aufgabe 4a
```

Inhalt der Datei **eingabe.txt**: **4 4 1 2 3 4 3 1 2 3 2 1 2 1 1**

Wie sieht die Ausgabe auf dem Bildschirm aus?

(b) Entwerfen Sie eine **statische Java-Methode insert**, die als Parameter ein **eindimensionales Array feld vom Typ int** und **zwei int-Variablen elem und pos** erwartet und selbst wieder ein eindimensionales **Array vom Typ int zurück** liefert.

Ziel der Methode ist das **Einfügen des Elements elem an der Stelle pos im Array feld und Rückgabe** des veränderten **Arrays**.

Das Array a ist bei Übergabe bereits vollständig gefüllt, die Länge von a ist beliebig. Der **Fehlerfall**, dass die Stelle c außerhalb des Array-Bereichs liegt, soll wie folgt abgefangen werden: In diesem Fall **soll das Array a unverändert zurück gegeben** werden.

(c) Implementieren Sie eine **statische Java-Methode reverse**, die genau **einen Parameter** vom Typ **eindimensionales Array vom Typ int** erwartet und als **Ausgabe ein eindimensionales Array vom Typ int zurückliefert**.

Interpretieren Sie dabei die Eingabe als eindimensionalen Vektor. Die Methode reverse soll diesen Vektor in umgekehrter Reihenfolge zurückgeben!
Beispiel: Falls als **Eingabe das Array {7, 3, 5, 4}** der Methode reverse übergeben wird, sollte das **Array {4, 5, 3, 7} ausgegeben** werden. Dies ist nur ein Beispiel. Die **Methode** muss **natürlich für Eingaben** (Vektoren) beliebiger Länge korrekt funktionieren.

4.)	a	b	c	Summe
Punkte	/13	/8	/8	/29

5. Aufgabe: Objektorientierung

Gegenstand dieser Aufgabe ist das Arbeiten mit Klassen und Objekten. Bei der Lösung der ersten Teilaufgabe achten Sie insbesondere darauf, dass Sie „ordentlich“ programmieren und insbesondere Einrückungen verwenden.

(a) Entwerfen Sie eine Klasse **QMatrix**, welche zur Darstellung und Bearbeitung von **quadratischen Matrizen der Form (n x n)** verwendet werden kann, entsprechend den folgenden Vorgaben an die Attribute, Konstruktoren und Methoden. Die **Matrix kann nur Werte vom Typ int** verwalten.

Als **Attribut** soll ein (quadratisches) **zwei-dimensionales Array feld** bereitgestellt werden, welches die quadratische Matrix der Größe n x n darstellt und von Außen nicht sichtbar ist.

Es soll genau ein **Konstruktor** mit **einem Parameter** vom Typ int bereitgestellt werden, welcher die Größe der Matrix erwartet und entsprechend Speicherplatz für das interne Array feld allokiert.

Insgesamt soll die Klasse QMatrix **3 öffentliche (**von Außen benutzbare**) Instanz-Methoden** anbieten: **fill**, **print** und **add**, die im folgenden näher beschrieben werden.

Die Methode **fill** erwartet einen Parameter vom Typ int und gibt keinen Wert zurück. Als Resultat sollen alle Werte der quadratischen Matrix (Array feld) mit dem Wert des Parameters aufgefüllt werden.

Die Methode **print** erwartet keinen Parameter und liefert auch keinen Wert zurück. Als Resultat soll der aktuelle Inhalt der Matrix (Array feld) auf der Standardausgabe ausgegeben werden. Dabei sollen die Werte einer Matrix-Zeile jeweils in einer neuen Zeile beginnen. Die Ausgabe jeder Zeile der Matrix muss also jeweils auf dem Display in einer neuen Zeile starten.

Die Methode **add** erwartet einen Parameter **par** vom Typ QMatrix und liefert einen Rückgabewert vom Typ QMatrix. Als Resultat soll ein neues Objekt **result** vom Typ QMatrix erzeugt werden. Der Inhalt der neuen Matrix **result** ergibt sich aus der Matrix-Addition des (die Methode) aufrufenden Objekts und des Parameter-Objekts **par**. Bei der Lösung können Sie davon ausgehen, dass die beiden zu addierenden (quadratischen) Matrizen von derselben Größe n sind. Ein entsprechender Fehlerfall muss also nicht abgefangen werden!

(b) Im folgenden seien die Java-Klassen **Alpha**, **Beta** und **Test** vorgegeben:

```
public class Alpha {
        protected String s;
        public Alpha (String s)  {this.s = s; }
        public String getS ( ) {return s;}
        public String toString ( )  {return "Alpha: " + s;}
} // Alpha

public class Beta extends Alpha {
        public Beta (String s) {super (s);}
        public String getS ( ) {return s;}
        public String toString ( ) {return super.toString ( ); }
} // Beta

public class Test {
        public static void main (String [] args) {
                Alpha a1 = new Alpha ("Alpha 1");
                Alpha a2 = new Beta ("Beta 1");
                System.out.println (a1.toString ());
                System.out.println (a2.toString ());
                System.out.println (a1.getS());
                System.out.println (a2.getS());
                System.out.println (((Beta) a2).getS());
        } // main
} // Test
```

Wie sieht die Ausgabe auf dem Bildschirm aus?

5.)	a	b	Summe
Punkte	/13	/7	/20