

## 1 Java-Programm: Methode implementieren (10 Punkte)

Gegeben sei eine Liste von 600 Klausurergebnissen (marks). Die Ergebnisse liegen zwischen 1.0 und 5.0.

Schreiben Sie in den folgenden drei Teilaufgabe je eine Java-Funktionen, indem Sie die vorgegebenen Methodenrumpfe ergänzen. Die Kommentierung der Methoden darf dabei in englischer oder deutscher Sprache erfolgen. Sie können davon ausgehen, dass jeweils ein korrektes Array mit 600 Einträgen übergeben wird.

Die Java-Funktionen sollen die folgenden Werte berechnen:

- a) die Durchschnittsnote (3 Punkte)

```
static double calcAverage(double[] marks){
    double result = 0.0;           // variable containing the result
    for (int i = 0; i < 600; i++){ // look at all marks in the array
        result = result + marks[i]; // add mark to result
    }
    return (result / 600);        // return the average mark
}
```

- b) die Durchfallquote in % (3 Punkte)

Ein Ergebnis > 4.0 bedeutet, dass der Teilnehmer durchgefallen ist.

```
static double calcFailureRate(double[] marks){
    int failureCounter = 0;       // counter for the failures
    for (int i = 0; i < 600; i++){ // look at all marks in the array
        if (marks[i] > 4.0) failureCounter++; // count mark if mark is 4.0 or worse
    }
    return ((double) failureCounter / 6.0); // return the failure rate as double in %
}
```

Hinweis:

return (failureCounter / 6.0); als vorletzte Zeile ist falsch, da dann eine Ganzzahldivision ausgeführt wird!

- c) Anzahl der Teilnehmer, die die beste Note aus der gegebenen Liste erreicht haben.  
(4 Punkte)

```
static int calcQuantityOfBestResults(double[] marks){
    int quantity = 0;           // variable containing the number of best marks
    double bestResult = marks[0]; // set the first mark as best mark
    for (int i = 0; i < 600; i++){ // look at all marks in the array
        if (marks[i] < bestResult){ // if the current mark is better than the best mark
            bestResult = marks[i]; // set it as the new best mark
            quantity = 1;
        }
        else{
            if (marks[i] == bestResult) { // otherwise if the current mark is the same as the best mark
                quantity++; // increase the quantity
            }
        }
    }
    return quantity; // return the quantity
}
```

Hinweis:

Die beste Note aus der Liste ist nicht zwangsläufig eine 1.0

## 2 Sprache und Grammatik (10 Punkte)

Die hawaiianische Sprache kennt nur die folgenden Buchstaben:

- die Vokale a, e, i, o, u
- die Konsonanten h, k, l, m, n, p, w

Es gelten dabei folgende Regeln:

Ein Wort beginnt mit einem Konsonanten oder einem Vokal. Auf einen Konsonanten muss mindestens ein Vokal folgen. Es können beliebig viele Vokale aufeinander folgen. Konsonanten dürfen nicht am Ende eines Wortes stehen. Ein Wort hat mindestens einen Buchstaben.

- a) Geben Sie eine CH-2 Grammatik an, die diese Sprache erzeugt. (3 Punkte)

**Hinweis:**

Es gibt auch CH-3 Grammatiken, aber diese sind recht umfangreich und daher als Lösung nicht sinnvoll.

$G = (\{a, e, i, o, u, h, k, l, m, n, p, w\}, \{S, V, K\}, R, S)$   
 $R = \{ S \rightarrow V|VS|KV|KVS$   
 $V \rightarrow a|e|i|o|u$   
 $K \rightarrow h|k|l|m|n|p|w \}$

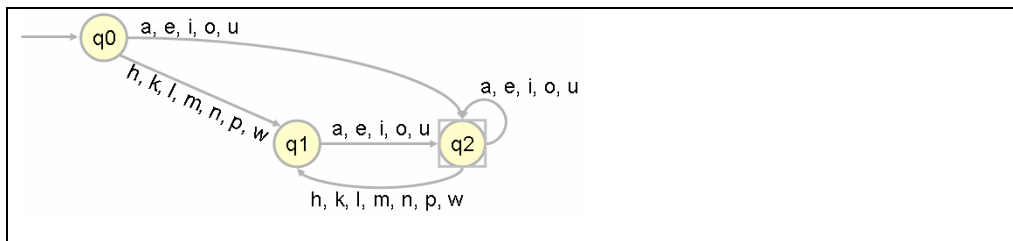
- b) Erzeugen Sie mittels Ihrer Grammatik aus a) das Wort kaiulani. (2 Punkte)

$S \rightarrow KVS \rightarrow kVS \rightarrow kaS \rightarrow kaVS \rightarrow kaiS \rightarrow kaiVS \rightarrow kaiuS \rightarrow kaiuKVS \rightarrow kaiuVS \rightarrow kaiulaS$   
 $\rightarrow kaiulaKV \rightarrow kaiulanV \rightarrow kaiulani$

- c) Erstellen Sie einen regulären Ausdruck für die hawaiianische Sprache. (2 Punkte)

$((a + e + i + o + u) + (h + k + l + m + n + p + w)(a + e + i + o + u))$   
 $((a + e + i + o + u) + (h + k + l + m + n + p + w)(a + e + i + o + u))^*$

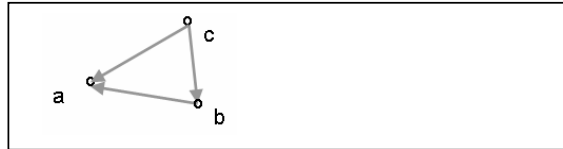
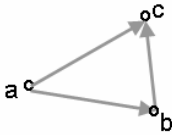
- d) Erstellen Sie einen Akzeptor, der die hawaiianische Sprache akzeptiert. (3 Punkte)



### 3 Graphen (10 Punkte)

- a) Erstellen Sie zu folgendem Graphen G1 den dualen Graphen. (1 Punkt)

G1:

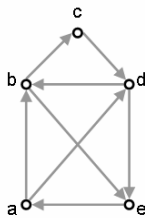


- b) Geben Sie für den Graphen G1 aus Teilaufgabe a) die Adjazenzmatrix an. (1 Punkt)

	a	b	c
a	0	1	1
b	0	0	1
c	0	0	0

- c) Geben Sie zu jeder Ecke des folgenden Graphen G2 den Ein- und Ausgangsgrad an. Verwenden Sie zur Lösung die rechts abgebildete Tabelle. (2 Punkte)

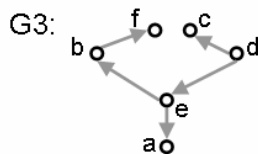
G2:



	$ \bullet e $	$ e \bullet $
a	1	2
b	2	2
c	1	1
d	2	2
e	2	1

- d) Stimmen die folgenden Aussagen für die Graphen aus Teilaufgabe a), c) und d)? (6 Punkte)

Verwenden Sie zur Antwort folgende Tabelle, indem Sie für ja ein J und für nein ein N eintragen. Pro fehlerhafter Antwort werden 0,25 Punkte abgezogen, fehlende Antworten mit 0 Punkten bewertet. Sollte daraus ein negatives Ergebnis für die Teilaufgabe d) resultieren, wird diese mit 0 Punkten bewertet.



Anmerkung: In der dritten Zeile werden bei J/N alle Lösungen als korrekt akzeptiert.

Aussage	G1	G2	G3
Der Graph enthält mind. einen Zyklus.	N	J	N
Der Graph ist ein gerichteter Baum.	N	N	J
Alle im Graph enthaltenen Zyklen sind einfache Zyklen.	J/N	N	J/N
Der Graph ist ein gerichteter Wald.	N	N	J
Der Graph enthält mind. einen Hamiltonschen Kreis.	N	J	N
Der Graph ist azyklisch.	J	N	J
Der Graph enthält mind. einen Eulerschen Zyklus.	N	N	N
Es gibt mind. eine Ecke für die gilt $ e \bullet  <  \bullet e $ .	J	J	J

## 4 Java-Programm: Verständnis (9 Punkte)

Gegeben sei die folgende Java-Klasse:

```
public class class1Program{
    static long x = 2;
    static long method1(long x, long y){
        /* 1 */
        if (y == 1) return x;
        else return x + method1(x, y - 1);
    }

    static long method2(long x, long y){
        /* 2 */
        if (y == 1) return x;
        else return method1(x, method2(x, y - 1));
    }

    public static void main(String[] args){
        long y = 5;
        /* 3 */
        Out.println(method2(x, y));
    }
}
```

- a) Was berechnen die beiden Methoden method1 und method2 und welche Ausgabe erzeugt das Programm? (5 Punkte)

method1(x, y) : Berechnet  $x * y$  (rekursiv)

method2(x, y) : Berechnet  $x^y$  (rekursiv unter Verwendung von method1 für die Multiplikation)

Ausgabe des Programms: 32

- b) Sind die Variablen an den Stelle /\* 1 \*/, /\* 2 \*/ und /\* 3 \*/ lebendig und/oder sichtbar? (4 Punkte)

Verwenden Sie zur Antwort folgende Tabellen, indem Sie für ja ein J und für nein ein N eintragen. Der Kontext, in dem die zu betrachtende Variable deklariert wurde, ist jeweils in Klammern angegeben. Pro fehlender oder fehlerhafter Antwort werden 0,5 Punkte abgezogen. Sollte daraus in der Summe ein negatives Ergebnis für die Teilaufgabe b) resultieren, wird diese mit 0 Punkten bewertet.

Lebendig:

Stelle	x (class1Program)	y (main)	x (method1)	y (method1)	x (method2)	y (method2)
/* 1 */	J	J	J	J	J	J
/* 2 */	J	J	N	N	J	J
/* 3 */	J	J	N	N	N	N

Sichtbar:

Stelle	x (class1Program)	y (main)	x (method1)	y (method1)	x (method2)	y (method2)
/* 1 */	N	N	J	J	N	N
/* 2 */	N	N	N	N	J	J
/* 3 */	J	J	N	N	N	N

## 5 Java-Programm: Fehler finden (8 Punkte)

Nachfolgendes Java-Programm soll die Quadratzahlen von 1 bis einschließlich 30 berechnen und ausgeben, sowie gleichzeitig die Summe über alle Quadratzahlen von 1 bis einschließlich 30 berechnen und am Ende ausgeben. Im Quelltext haben sich semantische und syntaktische Fehler eingeschlichen.

```
public class CalcQuadProgram {

    public void main (String[] args) {
        int result = 0;
        for (i = 1; i = 30; i++) {
            Out.println(i * i)
            result = result + i * i;
        }
        Out.println("result");
    }
}
```

- a) Schreiben Sie ein korrigiertes Programm, in dem Sie die korrigierten Fehler durch Kommentare aufzeigen. Das korrigierte Programm soll sich unter Java kompilieren und ausführen lassen sowie die oben definierten Aufgaben erfüllen. (4 Punkte)

```
public class CalcQuadProgram {
    public static void main (String[] args) {    // static was missing
        int result = 0;
        for (int i = 1; i <= 30; i++) {        // i wasn't declared and the condition was wrong
            Out.println(i*i);                  // missing semicolon
            result = result + i*i;
        }
        Out.println(result);                  // result should be written instead of 'result' to return the
                                                // correct value
    }
}                                             // missing closing bracket
```

**Formatiert:** Englisch  
(Großbritannien)

- b) Schreibe Sie ein neues Programm, das die Aufgaben mittels einer while-Schleife löst. Das neue Programm soll sich ebenfalls kompilieren und ausführen lassen. Zudem sollen die gleichen Variablennamen verwendet werden. (4 Punkte)

```
public class CalcQuadProgram {

    public static void main (String[] args) {
        int result = 0;                        // this variable will contain the sum of the square numbers
        int i = 1;                             // declare loop variable
        while (i <= 30) {                       // loop 30 times
            Out.println(i*i);                  // print i2
            result = result + i*i;             // increase the sum by the square number
            i++;                               // increase loop variable
        }
        Out.println(result);                  // print the sum of the square numbers
    }
}
```

## 6 Erweiterte Backus-Naur-Form (7 Punkte)

- a) Für einen Kunden soll eine Anwendung entwickelt werden, die u.a. Telefonnummern erfassen und auf Korrektheit überprüfen kann.

Sie hat folgenden Aufbau:

Sie beginnt durch ein Plus +. Danach folgt der Ländercode, der aus einer, zwei oder drei Ziffern besteht. Anschließend folgt die Vorwahl aus zwei bis vier Ziffern in Klammern. Anschließend folgt eine Rufnummer, die aus einer beliebigen Anzahl von Ziffern bestehen kann, aber mindestens aus einer Ziffer besteht. Wenn es sich bei dieser Rufnummer um einen Nebenstellenanschluss handelt, folgt nach den Ziffern für den Anschluss ein Bindestrich – und beliebig viele weitere Ziffern, wobei wieder mindestens eine Ziffer vorhanden sein muss.

### Hinweis:

Alle verwendeten Nichtterminal-Symbole sind zu definieren.

Beispiele für Telefonnummern:

+49(721)608-4046

+49(721)6084046

Erstellen Sie zur Überprüfung einer solchen Telefonnummer eine EBNF-Grammatik. (2 Punkte)

Ziffer = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0".  
 Telefonnummer = "+" Ziffer [Ziffer] [Ziffer] "(" Ziffer Ziffer [Ziffer] [Ziffer] ")" Ziffer {Ziffer} ["-" Ziffer {Ziffer}].

- b) Ergänzen Sie die unten angegebene EBNF-Grammatik, damit sie den korrekten Aufbau einer (statischen) Java-Methode beschreibt. Als Typen sind im Rahmen der Methodendeklaration nur int und float zugelassen. (5 Punkte)

Zur Darstellung eines Leerzeichens soll der Unterstrich \_ verwendet werden.

Die folgende Nichtterminalsymbole dürfen ohne Definition für Teilaufgabe b) verwendet werden. Alle weiteren verwendeten Nichtterminalsymbole sind zu definieren.

MName: Beschreibt den Aufbau zulässiger Methodennamen in Java.  
 VName: Beschreibt den Aufbau zulässiger Variablennamen in Java.  
 Block: Beschreibt den Rumpf einer Java-Methode inklusive der geschweiften Klammern.

Lösung1:

Method = "static\_" ("void" | Typ) "\_" MName "(" [Typ "\_" VName {"\_" Typ "\_" VName}] ")" Block.  
 Typ = "int" | "float" .

Alternative Lösung2:

Method = "static\_" ("void" | "int" | "float") "\_" MName "(" [{"int" | "float"} "\_" VName {"\_" ("int" | "float") "\_" VName}] ")" Block.

## 7 Wissensfragen (7 Punkte)

- a) Stimmen die folgenden Aussagen? (4 Punkte)  
Verwenden Sie für ja ein J und für nein ein N.

Korrekte Antworten werden mit 0,5 Punkten, falsche Antworten mit einem Abzug von 0,5 Punkten und nicht beantwortete Fragen mit 0 Punkten bewertet. Sollte daraus in der Summe ein negatives Ergebnis für die Teilaufgabe a) resultieren, wird die Teilaufgabe mit 0 Punkten bewertet.

Semantik beschreibt die Beziehungen, die sich unmittelbar aus der Anordnung der Zeichen ergeben.	N (das war Syntax)
Bei Java gilt für alle boolesche Variablen a,b: (a && b) == !(a    !b)	J
Es gibt genau einen Ausdruck in disjunktiver Normalform (DNF) zu einem gegebenen booleschen Ausdruck.	N (die Teilterme können umsortiert werden)
Eine Halbgruppe ist eine Algebra, die abgeschlossen ist und bei der das Kommutativgesetz gilt.	N (Assoziativgesetz gefordert)
Bei der theoretischen Effizienz gilt: $O(n^2) + O(n \log(n)) + O(n) = O(n^2)$	J
Zu jedem regulären Ausdruck gibt es äquivalente Akzeptoren.	J
Eine sichtbare Variable ist lebendig.	J
Die folgende Zeichenfolge ist ein korrekter EBNF-Ausdruck: wort = "a"("b c"){ "a" } "b" ["a"].	J

- b) An welchem Schlüsselwort erkennt man bei Java eine Prozedur und welchen Parameter muss es beschreiben? (1 Punkt)

void als Typ des Rückgabewerts

- c) Nennen Sie drei aus der Vorlesung bekannte Java-Datentypen. (1 Punkt)

Drei aus int, short, byte, char, float, double, String...

- d) Was bedeutet der Kleenesche Stern bei regulären Ausdrücken? (1 Punkt)

Die beliebig häufige ( $0..∞$ ) Wiederholung des (Teil-)Ausdrucks, auf den der Kleenesche Stern angewendet wird.