

Aufgabe 1 (Algebraische Strukturen) 1 Punkt

1.1 Fasst man ganze Zahlen mit gleichem Rest bei der Division durch $n \geq 2$ zu sogenannten Restklassen modulo n zusammen und bezeichnet die Restklasse einer Zahl $z \in \mathbb{Z}$ mit $[z]$, dann lassen sich zweistellige additive und multiplikative Verknüpfung von Restklassen durch die Addition und Multiplikation von beliebigen Elementen (Repräsentanten) dieser Klassen und anschließende Restbildung wie folgt definieren:

$$[z_1] + [z_2] := [z_1 + z_2]$$

$$[z_1] \cdot [z_2] := [z_1 \cdot z_2]$$

Auf diese Weise erhält man eine algebraische Struktur, die man mit $\mathbb{Z}/n\mathbb{Z}$ bezeichnet. Um welche Ihnen inzwischen bekannte algebraische Struktur handelt es sich dabei? Begründen Sie Ihre Antwort kurz.

Aufgabe 2 (Textverschlüsselung) 5 Punkte [2+2+1]

2.1 Eine einfache (aber nicht besonders sichere) Methode, um einen Text zu verschlüsseln, ist unter der Bezeichnung Cäsar-Code, Cäsarchiffre oder Verschiebchiffre bekannt. Dabei wird jeder Buchstabe des Alphabets um eine vorgegebene Anzahl an Stellen verschoben. Ein Beispiel:

Schlüssel=6
 Original: ABCDEFGHIJKLMNOPQRSTUVWXYZ
 Code: GHIJKLMNOPQRSTUVWXYZABCDEF

Beispiel
 Original: INFORMATIKISTTOLL
 Code: OTLUXSGZOQOYZZURR



Gaius Iulius Caesar

a) Schreiben Sie ein Programm das die Cäsarchiffre implementiert. Es soll ein String und der Verschiebefaktor eingelesen werden. Danach soll der Originalstring mit dem codierten String untereinander ausgegeben werden. Auf das i -te Zeichen eines Strings x kann mit $x(i)$ zugegriffen werden (erstes Zeichen ist $x(1)$). Beachten Sie dabei, dass bei folgender Implementierung:

```
subtype GrossBuchstaben is Character range 'A'..'Z';
codearray : array (GrossBuchstaben) of GrossBuchstaben;
```

GrossBuchstaben' Pos('A') nicht 0 sondern 65 ist!

b) Um einen Text sicherer zu verschlüsseln, verschiebt man die Buchstaben nicht immer um den gleichen Wert, sondern verwendet stattdessen einen zweiten Text und verschiebt immer um die Position des nächsten Buchstabens im zweiten Text. Ein Beispiel:

Original: INFORMATIKISTTOLL
 2.Text: H A L L O
 Position: 8-1-12-12-15 (und wieder von vorne)
 Code: QORAGUBFUZQTFDDTM

Schreiben Sie ein Programm, das diesen Verschlüsselungsalgorithmus realisiert.

c) Kodiert man die Buchstaben des Originaltexts und des mit Hilfe des Chiffrealphabets erzeugten Codes als ganze Zahlen – etwa 'A' := 0, ..., 'Z' := 25 – dann ist die Cäsarchiffre ein Spezialfall einer affinen Transformation über der algebraischen Struktur aus Aufgabe 1. Um welche affine Transformation handelt es sich dabei?

Aufgabe 3 (Newton-Verfahren)6 Punkte [0.5+0.5+3+2]

3.1 Das Newtonsche Näherungsverfahren ist ein Standardverfahren in der Mathematik, um die Nullstellen einer Funktion numerisch zu bestimmen. Ausgehend von einem Anfangswert x_0 , berechnet das Verfahren eine Folge von Werten $x_i, i = 1, \dots$ nach folgender Vorschrift:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Falls diese Folge zu einem Punkt konvergiert, stellt dieser Punkt eine Nullstelle der Funktion f dar.

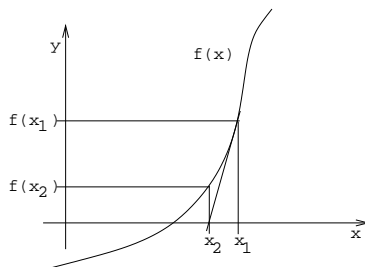
Gegeben sind die drei folgenden mathematischen Funktionen:

$$f_1(x) = -(x^4 - x^2) e^{-x^3}$$

$$f_2(x) = x^3 - 4x^2 - 2x + 4 + \frac{1}{x}$$

$$f_3(x) = \frac{1}{x^2} - 5 + \sqrt{|5x|}$$

Wieviele Nullstellen haben sie? (Es empfiehlt sich, den Graphen der Funktion zu skizzieren).



Sir Isaac Newton

Schreiben Sie jeweils eine Ada 95 Funktion für die Berechnung der Funktion $f_i(x)$ und deren Ableitung $f'_i(x)$ mit $i \in \{1, 2, 3\}$. Schreiben Sie ein Ada 95 Programm, das das Newton-Verfahren für diese drei Funktionen implementiert. Nach dem Starten des Programms soll der Startwert x_0 eingelesen werden. Danach wird die berechnete Nullstelle ausgegeben. Bei Funktionen mit mehreren Nullstellen bestimmt man diese durch das Starten des Programms mit verschiedenen Anfangswerten.

Abzugeben sind also:

- Die Graphen der Funktionen f_1 , f_2 und f_3 .
Es empfiehlt sich hier nicht von Hand gezeichnete Graphen einzuscannen. Stattdessen können Sie irgend ein Programm verwenden mit dessen Hilfe sich Graphen darstellen lassen.
- Die Anzahl der Nullstellen der Funktionen f_1 , f_2 und f_3 .
- Ein Ada 95 Programm für das Newton-Verfahren.
- Alle Nullstellen der Funktionen f_1 , f_2 und f_3 jeweils mit den Anfangswerten, die zu diesen Nullstellen geführt haben.

Hinweis: Sie müssen folgendes Paket in Ihr Programm einbinden, um mathematische Grundfunktionen verwenden zu können:

```
with Ada.Numerics.Elementary_Functions;  
use Ada.Numerics.Elementary_Functions;
```

Verwendungsbeispiel:

```
Y := Log(X, B);      --berechnet den Logarithmus von 'X' zur Basis 'B'  
Y := Exp(X);        --berechnet e hoch 'X'  
Y := Basis**Potenz; --berechnet 'Basis' hoch 'Potenz'
```

Aufgabe 4 (Backus-Naur-Form) 3 Punkte [1.5+0.5+1]

4.1 In einem alten Zauberbuch für Hexen ist folgende Definition für Zaubersprüche (in BNF) zu finden:

```

<Zauberspruch> ::= <H><E><X><E>
<H> ::= abra | sim | sam | som | sum | <X>
<E> ::= ca | da | bra | la | sa | so | dum | <X>
<X> ::= sim | sam | sum | la | dim | dam | dum | <E><E>
    
```

- a) Überprüfen Sie, ob folgende Zaubersprüche aus dem Zauberbuch stammen und begründen Sie Ihre Antwort. Antworten ohne Begründung werden nicht bewertet.
- | | |
|----------------------------|----------------------------|
| a.1) abracadasumsimsaladim | a.4) samcadadum |
| a.2) abradasum | a.5) abracadabrasomdaladum |
| a.3) sumsumsumbabrasimdum | a.6) lalalaladumsaladam |
- b) Wie lange sind die kürzesten Zaubersprüche, die man mit diesen Regeln konstruieren kann?
 c) Wie viele Zaubersprüche mit minimaler Buchstabenanzahl gibt es?

Aufgabe 5 (Rekursive und iterative Lösungen) 4 Punkte [2+2]

5.1 Erstellen Sie ein Ada 95 Programm, das eine Folge von Einzelzeichen einliest und in umgekehrter Reihenfolge wieder ausgibt. Das Ende der Eingabe soll durch \$ gekennzeichnet werden, es soll also z.B. für die Eingabe a b c \$ die Ausgabe \$ c b a erzeugt werden. Abzugeben sind **zwei** Varianten des Programms:

- a) Eine Variante die keine Rekursion verwendet
 b) Eine Variante die auf einer rekursiven Prozedur basiert

Aufgabe 6 (Formale Sprachen und die Kleeneschen Operatoren) 6 Punkte [3+3]

6.1 Formale Sprachen, also Mengen von Sequenzen über einem Alphabet Σ , treten sehr häufig in der Informatik auf. Besonders hilfreich sind in diesem Zusammenhang die Kleeneschen Operatoren, die wie folgt definiert sind:

$$\Sigma^* := \bigcup_{n=0}^{\infty} \Sigma^n \quad \text{sowie} \quad \Sigma^+ := \bigcup_{n=1}^{\infty} \Sigma^n$$

Dabei bezeichnet L^n die n -te Potenz der Menge L , also die Menge aller endlichen Sequenzen der Länge n über L . Für $L = \{x, y\}$ wäre zum Beispiel $L^2 = \{xx, xy, yx, yy\}$.

Gegeben seien die drei folgenden Alphabete:

$$\Sigma_1 = \{a, b\}, \Sigma_2 = \{a, b, c\}, \Sigma_3 = \{b, c\}$$

- a) Geben Sie für jede der folgenden Mengen die Anzahl der Elemente an und listen Sie alle Elemente auf.

1) $\Sigma_1 \Sigma_2 \Sigma_1$	2) $(\Sigma_1 \Sigma_3)^2$	3) $\Sigma_1^3 \cup \Sigma_2^2$
4) $\bigcup_{n=1}^2 \Sigma_2^n$	5) $\bigcup_{n=0}^3 \Sigma_1^n$	6) $(\Sigma_1 \setminus \Sigma_3) \cup (\Sigma_3 \setminus \Sigma_1)$

- b) Beschreiben Sie jede der folgenden Mengen **verbal**.

1) Σ_2^+	2) $\Sigma_1^* \setminus \left(\bigcup_{n=0}^3 \Sigma_1^n \right)$	3) $\Sigma_1^* \cap \Sigma_3^*$
4) $\Sigma_2^* \setminus \Sigma_1^*$	5) $\Sigma_2^* \setminus (\Sigma_1 \cup \Sigma_3)^+$	6) $\Sigma_2^* \setminus (\Sigma_1^+ \cup \Sigma_3^+)$