

Inhalt

Einführung in Scheme	2
Anleitung – Wie gehe ich mit einem Skript um?	3
Materialien zum Skript	7
Mit oder ohne Skript weitermachen	8
Modul 1 Bearbeitung von Listen in Scheme: first und rest.....	9
Kapitel 1 Listen in Scheme.....	9
Kapitel 2 Zusatz Was genau bedeutet first (eigentlich car genannt) und rest (eigentlich cdr genannt)?	11
Modul 2 Funktionen selber schreiben	13
<i>Modul 3 Rekursion</i>	15
Kapitel 1 einfache Rekursion	15
Kapitel 2 Endrekursion	17
Kapitel 3 Tipps und Tricks zum Schreiben rekursiver Funktionen	18

Einführung in Scheme

Ziele:

1. Mit Scheme als funktionaler Programmiersprache rekursive Funktionen programmieren können.

Kompetenzen, die am Ende erreicht sein müssen

Fülle folgenden Fragebogen zu Beginn des Themas und am Ende einmal aus:

Kompetenzen, die du nun erreicht haben solltest	+	0	-
Ich kenne grundlegende Konstrukte in Scheme und kann sie in einfachen Funktionen anwenden:			
- first/rest			
- cons			
- append			
- cond			
- Ergebnisraum			
Ich kann einfache Funktionen in Scheme selbst schreiben			
Ich verstehe, wie Rekursion funktioniert.			
Ich kann rekursive Funktionen in Scheme implementieren			
Ich kann gegebene Problemstellungen in formalisierter Umgangssprache zusammenfassen und dann in Scheme implementieren			

Anleitung – Wie gehe ich mit einem Skript um?

- Im Folgenden wird genau beschrieben, wie du mit einem Skript arbeiten kannst und worauf du dabei achten musst. Du kannst dir dies zusätzlich – oder stattdessen – auch in einem Video erklären lassen:
- Schau dir dazu auf www.youtube.com/alexkueck11 das Video mit dem Namen „Aufbau eines Skriptes“ an.
- Um dir das Thema „Kryptologie“ zu erarbeiten, kannst du grundsätzlich zwei verschiedene Wege gehen:
 - **Weg A** – Du bearbeitest **eigenständig die Projektaufgabe**. Dazu kannst du dir aus den Materialien raussuchen, was du gebrauchen kannst oder dir auch ganz eigene erklärende Materialien suchen.
 - **Weg B** – Du gehst die **einzelnen Module Schritt für Schritt** durch und erarbeitest dir so nach und nach die einzelnen Module (hier lernst du sozusagen Stück für Stück, wie die Projektaufgabe gelöst werden kann. Die Projektaufgabe wurde daher auch in „kleine Häppchen“ aufgeteilt und am Ende von jedem Modul wird ein Stück der Projektaufgabe gelöst).
- Mit beiden Wegen kannst du die geforderten Kompetenzen erwerben. Wenn du schon einiges über die funktionale Programmierung weisst und gerne an etwas knobelst, kannst du Weg A wählen. Behalte dabei aber immer auch im Auge, was du am Ende der Einheit können musst.
- Wenn du in diesem Bereich aber noch unsicher bist und das Thema lieber Schritt für Schritt erklärt bekommen möchtest, um es zu begreifen, wähle zunächst lieber Weg B. Auch hier löst du die Projektaufgabe, aber eben Schritt für Schritt und es wird dir vorgegeben, wie der Lösungsweg aussehen kann.
- Wenn du einen der beiden Wege eingeschlagen hast, bedeutet das allerdings nicht, dass du darauf festgelegt bist! Natürlich kannst du vom Projekt auch wieder auf die Module umsteigen, zum Beispiel, wenn du bei einer Sache nicht weiterkommst. Ebenso kannst du auch zur Projektaufgabe wechseln, wenn du nach ein paar Modulen merkst, dass du jetzt schon gut im Thema drin bist, und versuchen möchtest, eigenständig weiter zu knobeln.
- Lege dir eine Mappe an, in der du alle Lösungen und (Zwischen-) Ergebnisse zu den Aufgaben bzw. dem Projektvorhaben festhältst.

Wichtig: Du kannst deine Ergebnisse immer zwischendurch mit dem Lehrer abgleichen, um zu sehen, ob du auf dem richtigen Weg bist.

Gerade wenn du an dem Projekt arbeitest (aber auch wenn du mit dem Skript eigenverantwortlich durch das Thema gehst), ist es wichtig, dass du festhältst, wie du vorgegangen bist. Das tust du bitte in einem Blog oder einer Mappe. Dort hältst du fest, in welche Probleme du gelaufen bist und wie du sie gelöst hast – und vor allem, was du dadurch gelernt hast.

Wichtige Ergebnisse, Erkenntnisse, Merksätze oder Lösungsstrategien gehören hier ebenfalls hin. Am besten ist es, wenn du das in deinen eigenen Worten oder auch durch eine Skizze ausdrücken kannst. Selbst wenn das dann nicht ganz richtig ist, ist es besser so, als etwas Fertiges abzuschreiben. Der Lehrer kann so drauf schauen und dir helfen, wenn du etwas noch nicht vollständig verstanden hast.

Problemlösestrategien stehen bei diesem Projekt im Vordergrund nicht die Inhalte! Wenn du nicht genau weißt, was du aufschreiben sollst, lasse es dir vom Lehrer erläutern. Vorgefertigte Bögen, wo du Hilfestellung zu den Inhalten bekommst, kannst du beim Lehrer abholen.

Weg A – Bearbeitung der Projektaufgabe:

- Wie du die Projektaufgabe löst, bleibt dir und deiner Kreativität ganz allein überlassen. Du kannst selbst im Internet recherchieren und nachsehen, ob es dort Erklärungen oder Videos gibt, die dir weiterhelfen. Du kannst aber auch die in diesem Skript angegebenen Materialien weiter hinten verwenden – denn sie passen zu der Projektaufgabe.
- Ein Anhaltspunkt, um dich mit dem Thema auseinanderzusetzen, sind die Begriffe, die bei den zu erreichenden Kompetenzen am Anfang der Beschreibung angegeben sind. Damit könntest du z.B. anfangen. Wenn du die Begriffe verstehst und was für Ideen damit verknüpft sind, bist du meistens schon voll mit dem Thema beschäftigt. Vielleicht hast du ja auch schon für dich einen Weg gefunden, wie du an neue Projekte herangehst, dann solltest du diesen Weg hier auch gehen.
- Wichtig ist unbedingt, dass du im Auge behältst, was du am Ende der Einheit können musst. Die Projektaufgaben sind so formuliert, dass du am Ende alles, was gefordert ist, auch kannst. Aber es gibt ja viele Lösungswege und vielleicht kannst du am Ende das ein oder andere noch nicht so gut. Dann frage bitte nach zusätzlichen Übungsaufgaben zu dem jeweiligen Thema oder Begriff - bis du das Gefühl hast, es gut genug anwenden zu können.

Weg B – Bearbeitung der Module

- Gehe die Module Schritt für Schritt durch. Um die in jedem Modul angegebenen Aufgaben bearbeiten zu können, musst du vorher lernen, wie das geht. Nicht der Lehrer erklärt dir das, du entscheidest, auf welchem Weg du die Informationen haben möchtest und musst sie dann selbst so für dich zusammenstellen, dass du die Aufgaben lösen kannst. In der Regel kannst du wählen zwischen einem erklärenden Text, Webseiten, auf denen du passende Informationen findest oder erklärenden Videos. Diese kannst du dir so oft ansehen, wie du es brauchst und magst. Wenn du dennoch weitere Erklärungen benötigst, notiere dir deine Fragen und wende dich damit an deinen Lehrer oder suche im Internet selbst nach weiteren erklärenden Texten oder Videos. Der Lehrer ist da, um dich in deinem Lernen zu unterstützen, aber du musst aktiv werden und nachfragen, wenn etwas unklar ist.

- Es ist wichtig, dass du alle neuen Begriffe, die du nicht kennst, klärst und richtig verstehst. Du musst sie in eigenen Worten beschreiben oder sie in einer Skizze darstellen können.
- Gehe bei jedem der Kapitel wie folgt vor:
 1. Zu Beginn jedes Kapitels findest du Verweise auf Materialien, die dir helfen sollen, das Thema zu verstehen, damit du später die dazugehörigen Aufgaben lösen kannst. Das können zum Beispiel sein:
 - erklärende Videos,
 - Infotexte (parallel zu den Videos),
 - Seiten in einem Schulbuch und
 - Texte in Zeitschriften oder auch
 - Internetseiten
 2. Eventuell brauchst du trotz der Materialien eine zusätzliche Erklärung, dann frage beim Lehrer nach. Eventuell haben andere ja auch diese Frage, dann kannst du auch einen kurzen Lehrvortrag dazu bekommen oder ein zusätzliches erklärendes Video.
 3. Die Videos und Dateien, auf die in diesem Skript verwiesen werden, findest du
 - a. auf dem YouTube-Kanal: youtube.com/alexkueck11
Die Videos beginnen alle mit „Kryptologie-„
 4. Falls du in den Materialien auf unbekannte Begriffe stößt, notiere diese. Das können auch einfache Worte sein. Versuche sie mithilfe der weiteren Materialien oder durch eigene Recherchen zu klären.
 5. Wenn du das Thema verstanden hast und alle darin enthaltenen Fachbegriffe in deinen eigenen Worten oder mittels einer Skizze erklären kannst, gehst du weiter zu den Aufgaben:
 - Die Aufgaben fordern dich auf, das Gelernte nun anzuwenden.
 - Gehe die Aufgabe, die im Skript angegeben sind der Reihe nach durch (die Aufgaben sind logisch aufeinander aufgebaut, daher der Reihe nach durchgehen).
 - Wenn du eine Aufgabe nicht bearbeiten kannst, gehe noch einmal über die Materialien oder schaue dir das erklärende Video erneut an. Vielleicht hast du einen neuen Begriff oder eine neue Idee noch nicht ganz verstanden – dann hole das nun nach.
 - Wenn das nichts hilft, frage bei Mitschülern oder dem Lehrer nach. Lass dir aber nicht die ganze Aufgabe lösen. Wichtig ist, dass du eigenständig an einer Lösung arbeitest – auch wenn sie am Ende vielleicht nicht ganz richtig ist.
 - Wenn du an deiner Lösung zweifelst, schaue in den Musterlösungen nach (falls vorhanden) oder frage den Lehrer, ob er sich deine Ergebnisse auch zwischendurch anschauen kann.

Falls du bei einer Aufgabe doch noch Schwierigkeiten hast, schaue dir noch einmal die erklärenden Materialien an.

Wichtig ist, dass du selbst an den Lösungen arbeitest und nicht andere das machen lässt.

6. Wenn Aufgaben, die mit einem **Zusatz** gekennzeichnet sind, brauchst du nicht bearbeiten. Diese Aufgaben sind schwieriger und gehen über das hinaus, was du als Minimum erreichen musst, um das Skript erfolgreich abzuschließen. **Für eine abschließende Note im Einser- oder Zweier-Bereich solltest du aber zumindest einige dieser Zusatzaufgaben bearbeiten.**
7. Es wird zwischendurch Tests geben, diese werden rechtzeitig angegeben. Auch welche Kompetenzen in den Tests angefragt werden.

Wichtig:

Wichtig ist, dass du bei der Arbeit mit dem Skript selbst aktiv wirst und deinen eigenen Lernprozess überwachst:

- Liege ich noch gut in der Zeit?
- Habe ich alles verstanden/begriffen oder brauche ich noch Hilfe oder zusätzliche Erklärungen?
- Wie kann ich Zusammenhänge besser verstehen/begreifen, die noch unklar sind?
- Wer kann mir bei der Bearbeitung der Aufgaben helfen?

Du musst selbst entscheiden, wo du dir weitere Informationen/hilfen holen möchtest und von dir aus auf deinen Lehrer oder Mitschüler zugehen, um Fragen zu stellen, damit du die Themen und Begriffe besser verstehst und am Ende die geforderten Zielkompetenzen erreichst!

Es wird am Ende eine Klausur geben, die du bestehst, wenn du alle Aufgaben bearbeitet und verstanden/begriffen hast und die Kompetenzen erreicht hast.

Materialien zum Skript

Achtung:

Beim Schauen der Videos unbedingt Notizen machen. Haltet die Aussagen in eigenen Worten fest, die euch wichtig erscheinen!

Falls nach einmaligem Gucken die Anwendung schwer fällt, nehmt eine zum Video passende Aufgabe und löst sie parallel zum erneuten Gucken des Videos – Schritt für Schritt nach den Angaben im Video!

Videos

Die Videos sind alle auf dem YouTube-Kanal „**alexkueck11**“ zu finden und fangen fast alle mit “Kryptologie-” an.

<http://perm.ly/ki-scheme-editor>

<http://perm.ly/ki-scheme-konstanten>

<http://perm.ly/ki-scheme-first-rest>

<http://perm.ly/ki-scheme-listen-i>

<http://perm.ly/ki-scheme-listen-ii>

<http://perm.ly/ki-scheme-funktionen-selbst-schreiben>

<http://kkghh.de/neu.php?name=ki-scheme-funktionen-selbst-schreiben-II>

<http://kkghh.de/neu.php?name=ki-scheme-rekursion-einstieg>

<http://kkghh.de/neu.php?name=ki-rekursion-die-treppe>

<http://perm.ly/ki-scheme-rekursion-einfach-mit-aufgabe>

<http://perm.ly/ki-scheme-rekursion-mit-aufgabe-ii>

<http://perm.ly/ki-scheme-rekursion-am-beispiel>

<http://perm.ly/ki-scheme-hilfen-zu-funktionen>

<http://kkghh.de/neu.php?name=scheme-endrekursion>

<http://kkghh.de/neu.php?name=ki-scheme-rekursion-mit-zaehlen>

Texte

[Skript KI und Scheme -Text Editor und Lexikon](#)

[Skript KI und Scheme -Text first-rest-als Liste](#)

[Skript KI und Scheme -Text Einschub I Grundlagen in Scheme](#)

[Skript KI und Scheme -Text Funktionen selber schreiben](#)

[Skript KI und Scheme -Text Rekursion](#)

Mit oder ohne Skript weitermachen

Vorgehen:

Am Anfang des Skriptes wird dir eine **Projektaufgabe** gestellt.

1. du wechselst auf das Projekt, in dem du nur diese Aufgabe bearbeiten musst (behalte dabei im Auge, was du am Ende können musst - siehe "Kompetenzen")!
2. In diesem Skript wird das Projekt Stück für Stück erarbeitet und du bekommst detaillierte Erklärungen und einen Weg, wie du vorgehen musst, vorgegeben. Es gibt dir gezielt Hilfestellung mit erklärenden Texten und Videos, um die Projektaufgabe Stück für Stück zu lösen.

Wenn du unsicher bist, welchen Weg du einschlagen sollst, komme zu mir und wir klären das.

Wichtig: Du kannst deine Ergebnisse immer zwischendurch mit dem Lehrer abgleichen, um zu sehen, ob du auf dem richtigen Weg bist, aber du musst in einem Blog oder einer Mappe dokumentieren, in welche Probleme du gelaufen bist und wie du sie gelöst hast – und vor allem, was du dadurch gelernt hast. Problemlösestrategien stehen bei diesem Projekt im Vordergrund nicht die Inhalte!

Modul 1 Bearbeitung von Listen in Scheme: first und rest

Ein Verschlüsselungssystem benötigt eine Zuordnung von Buchstaben und Zahlen, da Verschlüsselungen in der Regel mathematische Operationen beinhalten:

- Das könnte folgendermaßen aussehen:
$$\begin{array}{lcl} a & - & 0 \\ b & - & 1 \\ & & \dots \end{array}$$
- Welche Operationen auf den Zahlen ausgeführt werden, dazu später mehr!

Wir wollen zunächst eine ganz einfache Verschlüsselung in **SCHEME** erstellen.

Kapitel 1 Listen in Scheme

Wir fangen erst mal mit Alphabet an. Wie man das in Scheme umsetzt - lies dazu den unten angegebenen Text oder schaue die Videos.

➔ **Video:**

- <http://perm.ly/ki-scheme-editor>
- <http://perm.ly/ki-scheme-konstanten>
- <http://perm.ly/ki-scheme-first-rest>
- <http://perm.ly/ki-scheme-listen-i>
- <http://perm.ly/ki-scheme-listen-ii>

○

➔ **Eigenständige Recherche im Internet:**

○ ...

➔ **Erklärender Text:**

- [Skript KI und Scheme -Text Editor und Lexikon](#)

Dann kannst du die folgenden Aufgaben bearbeiten.

Hier eine Beispiel-Liste (im oberen Fenster eingeben):

```
(define *alphabet*  
' ( (a 0)  
      (b 1)  
      (c 2)  
      (d 3)  
      (e 4)  
      (f 5) ) )
```


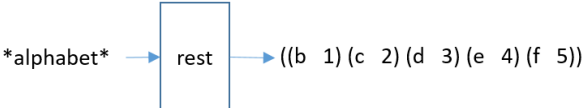
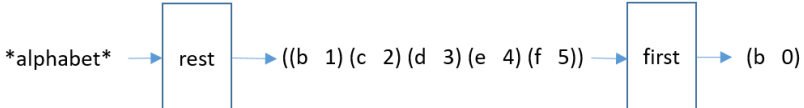
Gib die folgenden Ausdrücke im Interpreterfenster (unteres Fenster) ein und notiere die Antworten:

Eingabe	Ausgabe
(first *alphabet*)	
(rest *alphabet*)	
(first (rest *alphabet*))	
(first (first *alphabet*))	
(rest (first *alphabet*))	
(first (rest (first *alphabet*)))	

Beschreibe, was die Funktionen **first** und **rest** machen:

Stellt man jeden Funktionsaufruf durch eine Box dar, so können die obigen Hintereinanderausführungen der Funktionen **first** und **rest** wie folgt dargestellt werden (beachte, dass verschachtelte Funktionen immer von innen nach außen bearbeitet werden):

(<http://perm.ly/ki-scheme-first-rest>)

Eingabe	Darstellung
(first *alphabet*)	
(rest *alphabet*)	
(first (rest *alphabet*))	

Aufgabe 1.1:

Ergänze die Darstellungen für die restlichen Funktionsaufrufe:

<code>(first (first *alphabet*))</code>	
<code>(rest (first *alphabet *))</code>	
<code>(first (rest (first *alphabet *)))</code>	

Zusätzlich: Angenommen, folgende Liste sei gegeben:

li entspricht ``((1 2 3) (4 5 6) (7 8 9))`

- i. Gib an, welche Ausgaben durch folgende verkettete Funktionen erfolgen würden:
 - a. `(first (rest (rest (first `(1 2 3) (4 5 6) (7 8 9))))))`
 - b. `(rest (first (rest `(1 2 3) (4 5 6) (7 8 9))))`
 - c. `(first (first (rest `(1 2 3) (4 5 6) (7 8 9))))`
- ii. Gib an durch welche Verkettung der Funktionen „first“ und „rest“ du folgende Ausgaben erhalten würdest:
 - a. 2
 - b. 6
 - c. (2 3)

Kapitel 2 **Zusatz** Was genau bedeutet **first** (eigentlich car genannt) und **rest** (eigentlich cdr genannt)?

→ **Erklärender Text:**

- Skript KI und Scheme -Text first-rest-als Liste

Kapitel 3 Kompetenzen

Aufgabe 1.3.

Überprüfe zu deiner Sicherheit, ob du folgende Dinge nun kannst (fordere gegebenenfalls weitere Aufgaben zum Üben beim Lehrer an):

Kompetenzen, die du nun erreicht haben solltest	+	0	-
Ich kann mit dem Editor Dr. Racket (für die Implementierung von Scheme-Funktionen) umgehen.			
Ich kenne die polnische Schreibweise und kann sie in Scheme-Funktionen umsetzen.			
Ich verstehe, dass man in Scheme statt mit Variablen mit Listen arbeitet.			
Ich kann Listen in Scheme anlegen und sie mit <code>first</code> und <code>rest</code> bearbeiten.			
Ich kann Konstanten in Scheme anlegen und verstehe ihren Einsatzzweck.			
Ich kenne grundlegende Konstrukte in Scheme und kann sie in einfachen Funktionen anwenden:			
- <code>first/rest</code>			

Modul 2 Funktionen selber schreiben

Hier sollst du erste kleinere Funktionen selbst schreiben. Verwende dazu grundlegende Funktionen, die Scheme bereits vorgibt (first, rest, cons, append, cond)

→ **Video:**

- <http://perm.ly/ki-scheme-funktionen-selbst-schreiben>
- <http://kkghh.de/neu.php?name=ki-scheme-funktionen-selbst-schreiben-II>
- <http://kkghh.de/neu.php?name=ki-scheme-funktionen-selbst-schreiben-III>

→ **Eigenständige Recherche im Internet:**

- ...

→ **Erklärender Text:**

- [Skript KI und Scheme -Text Funktionen selber schreiben](#)

Aufgabe 2.1:

Schreibe nach dem Muster der Funktion **zweites** ebenso die Funktionen

- a) **drittes** (mit Probe, ob die Liste überhaupt lang genug ist!)
- b) eine Funktion **ohneErstes** (die Funktion gibt den Rest der Liste ohne das erste Element zurück).
- c) eine Funktion **ohneErstesundZweites** (die Funktion gibt den Rest der Liste ohne das erste und zweite Element zurück).
- d) Eine Funktion **ohneZweites** (die Funktion gibt den Rest der Liste ohne das zweite Element zurück).
- e) Eine Funktion **ohneDrittes** (die Funktion gibt den Rest der Liste ohne das dritte Element zurück).

- **Tipp:** benutze für d) und e) die Scheme-Funktion **cons**, **append** bzw. **list** um Listen zusammenzusetzen. Siehe dazu **Video** <http://perm.ly/ki-scheme-funktionen-selbst-schreiben-iii>

- c) **Zusätzlich:** Probiere auch, eine Funktion **n-tes** zu schreiben.

Tipp: Die Funktion **n-tes** muss offensichtlich zwei Parameter haben, nämlich die Zahl **n** und die **liste** (also (**define** (**n-tes** **n liste**), dessen **n-tes** Element zurückgegeben werden soll. Ein Problem besteht darin, dass es von dem Wert des Parameters **n** abhängt, wie oft eine Restliste gebildet werden muss. Dies ist zum Zeitpunkt der Funktionsdefinition unbekannt. Eine mögliche funktionale Modellierung lautet:

Für den Fall, dass

1. *die Liste leer ist, gib zurück: „die liste enthaelt nicht genug elemente“;*
2. *die Zahl $n \leq 0$ ist, hat der Benutzer der Funktion einen unsinnigen Parameter angegeben. Er erhält einen entsprechenden Hinweis.*
3. *die Zahl $n = 1$ ist, wird das erste Element der Liste zurückgegeben.*
4. *andernfalls wird das $n-1$ -te Element der Restliste bestimmt (Achtung Rekursion!!!!).*

Aufgabe 2.2.

Überprüfe zu deiner Sicherheit, ob du folgende Dinge nun kannst (fordere gegebenenfalls weitere Aufgaben zum Üben beim Lehrer an):

Kompetenzen, die du nun erreicht haben solltest	+	0	-
Ich kenne grundlegende Konstrukte in Scheme und kann sie in einfachen Funktionen anwenden:			
- cons			
- append			
- cond			
Ich kann einfache Funktionen in Scheme selbst schreiben			

Modul 3 Rekursion

Kapitel 1 einfache Rekursion

→ Video:

- <http://kkghh.de/neu.php?name=ki-scheme-rekursion-einstieg>
- (Alt: <http://kkghh.de/neu.php?name=ki-scheme-rekursion-die-treppe>)
- <http://perm.ly/ki-scheme-rekursion-einfach-mit-aufgabe>
- <http://perm.ly/ki-scheme-rekursion-mit-aufgabe-ii>
- <http://perm.ly/ki-scheme-rekursion-am-beispiel>
- <http://perm.ly/ki-scheme-hilfen-zu-funktionen>
- <http://perm.ly/ki-scheme-endrekursion>

→ Eigenständige Recherche im Internet:

- ...

→ Erklärender Text:

- [Skript KI und Scheme -Text Rekursion](#)

Problemstellungen werden in Scheme rekursiv gelöst und nicht mit Schleifen. Rekursion ist dem menschlichen Denken nachempfunden und bietet sich bei der Verschachtelung von Funktionen, die in Scheme betrachtet wird, daher besonders an.

Was bedeutet Rekursion?

1. Eine sich selbst aufrufende Funktion.
2. Es muss eine Abbruchbedingung existieren, die das „sich selbst aufrufen beendet, wenn sie erreicht wurde.“
3. Die an die Funktion übergebenen Parameter müssen bei jedem erneuten Aufruf so gestaltet sein, dass sich das ursprüngliche Problem verringert und irgendwann die Abbruchbedingung erreicht werden kann.

Voraussetzung: Zerlegung einer komplexen Problemstellung in kleinere Probleme muss möglich sein.

Das Schreiben rekursiver Funktionen erfordert einige Übung. Daher die folgenden Übungsaufgaben:

Aufgabe 3.1: Lösungsvideo: „KI – Rekursion II “ - <http://youtu.be/5B2KqSXOfA4>

Entwickle jeweils eine funktionale Modellierung, d.h.

1. formuliere einen Algorithmus in formalisierter Umgangssprache.
2. implementiere diese anschließend mit SCHEME und
3. zeichne jeweils einen Rekursionsbaum.

Verwende das Scheme-Konstrukt „**cons**“ oder „**append**“, um z.B. die Elemente einer Liste, die nicht entfernt werden sollen, „zwischen zu speichern“ (siehe Scheme-Kurzreferenz)), um sie anschließend (nach dem rekursiven Aufstieg) ausgeben zu können.

- a) Die Funktion **entfernen1** soll, wenn sie ein **objekt** in einer Liste **liste** gefunden hat, dieses anschließend ausgeben. Diesmal soll die Funktion aber für eine Lexikon-Liste wie oben beschrieben funktionieren (Bsp: (entfernen 3 `(1 2 3 4 5) -> 3))

- b) Die Funktion **enthalten?** soll prüfen, ob ein **objekt** in einer **liste** enthalten ist. Falls ja, soll sie **#t** (true) zurückgeben, andernfalls **#f** (false)
Bsp: (enthalten? 3 '(1 2 3 4)) -> #t.
- c) Die Funktion **enthalten?** soll prüfen, ob ein **Buchstabe (char)** in einem **String** enthalten ist. Falls ja, soll sie **#t** (true) zurückgeben, andernfalls **#f** (false)
Bsp: (enthalten? 'b '(„abcd“)) -> #t.
Tipp: es gibt in Scheme die Funktion **string->list**, welche aus einem String eine Liste aus einzelnen Buchstaben macht (siehe Kurzreferenz Scheme).
Beachte: ein einzelner Buchstabe wird in Scheme folgendermaßen dargestellt: #\a
- d) Die Funktion **entfernen2** soll die Liste zurückgeben, die entsteht, wenn man alle Vorkommen von **objekt** aus einer vorgegebenen **liste** entfernt.
Bsp: (entfernen2 3 '(1 2 3 4 3 5)) -> (1 2 4 5)
- e) Die Funktion **ersetzen** soll die Liste zurückgeben, die entsteht, wenn man alle Vorkommen von **objekt1** in einer vorgegebenen **liste** durch **objekt2** ersetzt.
- f) Die Funktion **verschieben** soll die Buchstaben einer Liste im Alphabet um jeweils 1 Stelle verschieben: '(a b c d) -> '(b c d e).
Tipp: es gibt in Scheme die Funktionen **char->integer** und **integer->char**, welche aus Buchstaben Zahlen machen bzw. umgekehrt.
Zusatz: Schreibe die Funktion so um, dass die Verschiebung beliebig sein kann 1,2,3,4... Überlege dir dann auch, was du tun kannst, wenn z.B. ein z um 3 verschoben wird (Tipp: Modulo Rechnung)
- g) **Erstelle ein kurzes Video zu einer der oben aufgeführten Funktionen, indem du deine Überlegungen und Implementierungen dazu erläuterst. Füge dies deinem Portfolio hinzu.**

Kapitel 2 Endrekursion

→ Video:

- <http://kkghh.de/neu.php?name=scheme-endrekursion>
- <http://perm.ly/ki-scheme-endrekursion>

→ Erklärender Text:

- [Skript KI und Scheme -Text Rekursion \(letzte Seite\)](#)

Um den speicheraufwendigen rekursiven Aufstieg zu vermeiden, gibt es die Möglichkeit eine sogenannte Endrekursion zu programmieren. Zwar gibt es hier auch den rekursiven Abstieg durch den wiederholten rekursiven Aufruf, jedoch wird jeder Aufruf der Funktion vollständig bearbeitet, so dass nichts „zwischen gespeichert“ werden muss und somit auch nicht beim rekursiven Aufstieg wieder zusammengesetzt werden muss. Der Compiler kann die Rekursion dann intern als „Iteration“ behandeln und ist dadurch schneller.

Aufgabe 3.2.

Schreibe folgende Funktionen so um, dass sie mittels Endrekursion arbeitet:

- Die Funktion **entfernen** soll die Liste zurückgeben, die entsteht, wenn man alle Vorkommen von **objekt** aus einer vorgegebenen **liste** entfernt (Signatur der Funktion sieht bei Endrekursivem Ablauf z.B. so aus: (entfernen objekt liste ausgabe) – wobei ausgabe der Parameter ist, in dem bei jedem rekursiven Schritt die Ausgabefunktion bereits zusammengestellt wird, so dass die Ausgabe bereits am Ende des rekursiven Abstiegs dort enthalten ist)

-Die Funktion entfernen3 soll aus einer Liste lia alle Elemente, die dort Vorkommen in einer zweiten Liste lib entfernen und anschließend lib ohne diese Elemente ausgeben (Signatur der Funktion sieht bei Endrekursivem Ablauf z.B. so aus: (entfernen3 lia lib ausgabe) – wobei ausgabe der Parameter ist, in dem bei jedem rekursiven Schritt die Ausgabefunktion bereits zusammengestellt wird, so dass die Ausgabe bereits am Ende des rekursiven Abstiegs dort enthalten ist).

Kapitel 3 Tipps und Tricks zum Schreiben rekursiver Funktionen

Man kann Funktionen häufig auf mehrere Arten schreiben. Wichtig dabei ist, einige „Tricks“ zum Schreiben rekursiver Funktionen zu beherrschen und dann diejenige Lösung zu wählen, die am schnellsten zum Ziel führt! Diese „Tricks“ sollen in der nächsten Aufgabe jeweils angewendet werden, um die geforderte Funktion zu implementieren (auch wenn man die Funktion anders ebenfalls implementieren könnte).

Aufgabe 3.3.

Funktion 1: (ersetzen elem1 elem2 liste)

Ersetze ein Element einer Liste durch ein anderes

- **Trick:** mit cons Zwischenwerte speichern

Funktion 2: (zaehlen elem liste zaehler)

Zähle die Vorkommen eines Elementes in einer Liste

- **Trick:** einen zusätzlichen Parameter einbauen, der zählt...
- <http://kkghh.de/neu.php?name=ki-scheme-rekursion-mit-zaehlen>

Funktion 3: (addieren la addListe backup ausgabe)

Addiert zu jedem Element einer ersten Liste (la) einen Wert aus der zweiten Liste (addListe). Ist die zweite Liste kürzer als die erste, so sollen die zu addierenden Werte dieser zweiten Liste (addListe) einfach solange wiederholt werden, bis die erste Liste komplett abgearbeitet ist. Bsp.: (addieren '(1 2 3 4 5 6 7 8) '(1 2) '(1 2) '()) -> (2 4 4 6 6 8 8 10)

Beachte: Es darf keine Subfunktion verwendet werden!

- **Trick:** eine Art Backup-Parameter einfügen, in dem der ursprüngliche Wert erhalten bleibt und nicht durch Rekursion in Richtung Abbruchbedingung verändert wird.

Funktion 4: (entfernen elem liste ausgabe)

Gib alle Elemente aus, die in der Liste vorkommen außer dem zu entfernenden Element elem

- **Trick:** Endrekursion – einen zusätzlichen Parameter einbauen, in dem das Ergebnis schon beim rekursiven Abstieg zusammengesetzt wird)

Funktion 5: (suche la lb zaehler)

alle Vorkommen der Elemente aus Liste la werden in Liste lb gesucht und in zaehler gezählt (mit einer Unterfunktion lösen)

- **Trick:** Subfunktionen verwenden
- <http://kkghh.de/neu.php?name=ki-scheme-rekursion-mit-unterfunktion>

Zusätzlich:

- a) Die Funktion **verketteten** soll eine Liste aller Elemente von **liste1** und **liste2** zurückgeben. Die Reihenfolge der Listenelemente soll unverändert bleiben. Bsp.: (verketteten '(a b c) '(1 2 3)) liefert als Ergebnis '(a 1 b 2 c 3).
- b) Die Funktion **umdrehen** soll eine Liste zurückgeben, die die Elemente einer vorgegebenen **liste** in umgekehrter Reihenfolge enthält.
- c) Die Funktion (loeschen la lb **zwischenListe** ausgabe) soll alle Elemente einer ersten Liste (la) in einer zweiten Liste (lb) löschen. Dazu soll aber **keine** Unterfunktion verwendet werden.

Trick: es wird ein Parameter **zwischenListe** hinzugefügt, in dem die Liste lb nach jeweils einem Durchlauf des Löschens mit einem Wert von Liste la zwischengespeichert wird. Diese Liste wird dann im nächsten Durchlauf als neuer Wert für Liste lb verwendet.

Kapitel 4 Tipps und Tricks zum Schreiben rekursiver Funktionen

Nun ein paar Vorübungen zur Kryptologie. Es geht darum Buchstaben in Zahlen zu verwandeln.

Aufgabe 3.4.

Funktion 1: (addiere buchstabe1 buchstabe2 liste)

In der Alphabetliste sollen die Zahlen zu den zwei gegebenen Buchstaben rausgesucht und addiert werden.

Bsp.: (addiere 'b 'c *alphabet*) -> 3

Verwende dabei folgende Konstante als Alphabetliste:

```
(define *alphabet*
  '( (a 0)
      (b 1)
      (c 2)
      (d 3)
      (e 4)
      (f 5)
      (g 5)
      (h 5)
      (i 5)
      (j 5) ))
```

Funktion 2: (addiere2 buchstabe1 buchstabe2 liste)

In der Alphabetliste sollen die Zahlen zu den zwei gegebenen Buchstaben rausgesucht und addiert werden. Anschließend soll der Buchstabe aus der Liste rausgesucht werden, der der Summe entspricht. Diese soll ausgegeben werden.

Bsp.: (addiere2 'b 'c *alphabet*) -> d

Zusatz: Überlege dir eine Lösung, falls die Summe eine Zahl ergibt, die größer ist als die größte in der Alphabetliste vorkommende Zahl – Tipp: modulo-Rechnung!!!!

Aufgabe 3.5.

Überprüfe zu deiner Sicherheit, ob du folgende Dinge nun kannst (fordere gegebenenfalls weitere Aufgaben zum Üben beim Lehrer an):

Kompetenzen, die du nun erreicht haben solltest	+	0	-
Ich verstehe, wie Rekursion funktioniert.			
Ich kann rekursive Funktionen in Scheme implementieren			