**Team**: < 04 >, < Schäfer & Ahmad >

**Aufgabenaufteilung**:

1. <Aufgaben, für die Teammitglied 1 verantwortlich ist>,   
   <Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden>
2. <Aufgaben, für die Teammitglied 2 verantwortlich ist>,   
   <Dateien, die komplett/zum Teil von Teammitglied 2 implementiert/bearbeitet wurden>

**Quellenangaben**: <

<http://users.informatik.haw-hamburg.de/~klauck/AlguDat/aufg1.html>

<http://users.informatik.haw-hamburg.de/~klauck/AlguDat/AD.pdf>

>

**Begründung für Codeübernahme**: < Es wurde kein Code übernommen. >

**Bearbeitungszeitraum**: <

Für den Entwurf: 2 Stunden 12.10

>

**Aktueller Stand**: < >

**Änderungen im Entwurf**: < KEINE >

Es sollen folgende ADTs in Erlang/OTP implementiert werden und die Dateien müssen wie vorgegeben heißen:

* Liste („liste.erl“)
* Stack („stack.erl“)
* Queue („schlange.erl“)
* Array („arrayS.erl“)

Es folgen nun erst einmal die Signaturen der ADTs. Bei denen ist es sehr wichtig, dass diese exakt eingehalten werden, um ggf. ADTs austauchbar mit anderen Gruppen zu machen, die sich ebenfalls an diese Signaturen halten.

**ADT Liste**  
Zu realisieren als einfach verkettete Liste.

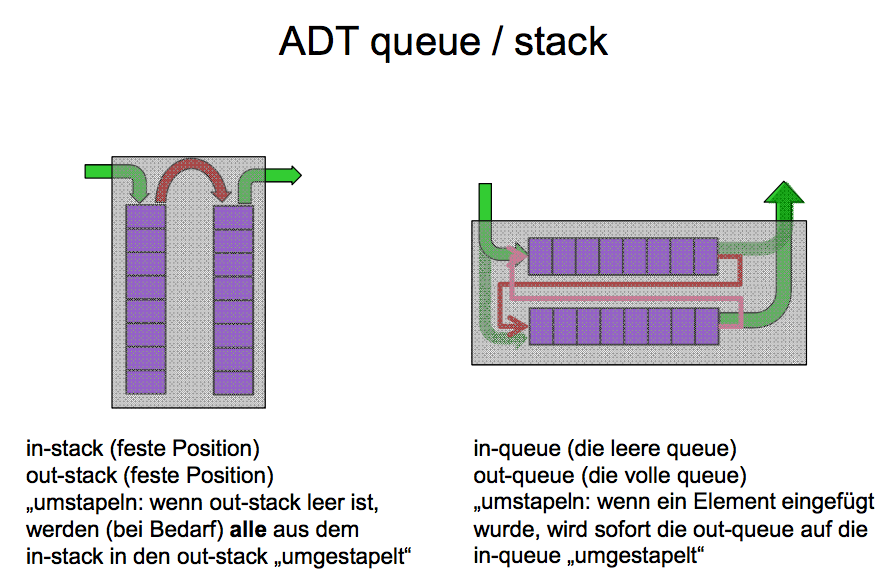
create:     ∅ → list  
isEmpty:   list → bool   
laenge:    list → int  
insert:      list × pos × elem → list  
delete:      list × pos → list  
find:      list × elem → pos          
retrieve:   list × pos → elem  
concat:     list × list → list

Weitere Vorgaben:

* Eine ADT Liste ist zu implementieren: als einfach verkette Liste
* Die Liste beginnt bei Position 1
* Die Liste arbeitet nicht destruktiv, d.h. wird ein Element an einer vorhandenen Position eingefügt, wird das dort stehende Element um eine Position verschoben

**ADT Stack**  
Zu implementieren auf der ADT Liste, soll heißen nur die Funktionen der ADT Liste dürfen intern verwendet werden um mit dem Stack zu arbeiten.

createS: ∅ → stack  
push: stack × elem → stack  
pop: stack → stack  
top: stack → elem  
isEmptyS: stack → bool

**ADT Queue**  
Zu implementieren auf der ADT Stack, mit zwei expliziten Stacks.  
Ein „In-Stack“ und ein „Out-Stack“, die gemeinsam eine Schlange realisieren sollen.

createQ: ∅ → queue  
front: queue → elem (Selektor)  
enqueue: queue × elem → queue  
dequeue: queue → queue (Mutator)  
isEmptyQ: queue → bool

**ADT Array**Zu implementieren auf der ADT Liste.

initA: ∅ → array  
setA: array × pos × elem → array  
getA: array × pos → elem  
lengthA: array → pos

Weitere Vorgaben:

* Das Array beginnt bei Position 0
* Das Array arbeitet destruktiv, d.h. wird ein Element an einer vorhandenen Position eingefügt, wird das dort stehende Element überschrieben
* Die Länge des Arrays wird bestimmt durch die bis zur aktuellen Abfrage größten vorhandenen und explizit beschriebenen Position im array
* Das Array ist mit 0 initialisiert, d.h. greift man auf eine bisher noch nicht beschriebene Position im Array zu erhält man 0 als Wert
* Das Array hat keine Größenbeschränkung, d.h. bei der Initialisierung wird keine Größe vorgegeben

**Fehlerbehandlung**  
Sollten nicht vorhandene Elemente gelöscht werden, in der Liste an unmöglicher Stelle eingefügt werden, von einem leeren Stack das oberste Element gelöscht werden etc. ist die Fehlerbehandlung durch "Ignorieren" durchzuführen, d.h. es wird so gehandelt, als wäre die Operation in Ordnung gewesen und z.B. eine nicht modifizierte Datenstruktur zurück gegeben.

**Wichtig**Es muss eine Struktur für die einfach verkettete Liste gebaut werden, die die Anforderungen erfüllt.Es muss entschieden werden, wie man damit umgeht, dass die ADT Liste mit Position 1 beginnt und die ADT Array, die auf dieser aufbauen soll, mit Position 0.