1 Aufgabe Allgemeine Fragen.

1. Womit kann man einen Boolschen Ausdruck im allgemeinen nicht in seine minimale Form umwandeln?

   ☐ Ablesen aus Wahrheitstabelle.
   ☐ Verfahren von Quine und McCluskey.
   ☐ KV-Diagramme.
   ☐ Anwendung boolescher Axiome.

2. Worin unterscheidet sich die doppelt verkettete Liste im Vergleich zur einfach verketteten Liste?

   ☐ Jeder Knoten zeigt auf seine zwei nachfolgenden Knoten.
   ☐ Jeder Knoten speichert zusätzlich zu seinem eigenen Wert auch den Wert des Vorgängers.
   ☐ Sie unterscheidet sich lediglich in einer Tail-Referenz, die auf das letzte Element der Liste zeigt.
   ☐ Jeder Knoten zeigt auf seinen Vorgänger und Nachfolger.

3. Welchen Aufwand hat die Binäre Suche?

4. Was gilt in einem AVL-Baum?

5. welches Verfahren sucht den kürzesten Weg in einem Graphen?

6.

2 Boolesche Algebra.

2.1 Teilaufgabe: Sind die folgenden Booleschen Ausdrucke äquivalent? Überprüfen Sie die Äquivalenz anhand der Wahrheitstafelmethode.

$f(x,y,z) = \overline{x \oplus y} + x \cdot z + z$
$g(x,y,z) = (\overline{x} + y) \cdot (x + \overline{y}) + z$

2.2 Teilaufgabe: Sind die folgenden Booleschen Ausdrucke äquivalent? Überprüfen Sie die Äquivalenz anhand der Algebraischen Umformung.

$f(x,y,z) = \overline{x \oplus y} + x \cdot z + z$
$g(x,y,z) = (\overline{x} + y) \cdot (x + \overline{y}) + z$

2.3 Wandeln Sie den folgenden Booleschen Ausdruck mit Hilfe der KV-Tafeln in eine minimale disjunktive Normalform um. Die Zwischenschritte müssen erkennbar sein.

$f(w,x,y,z) = (\overline{w} + x + z) \cdot (\overline{w} + x + \overline{z}) \cdot (\overline{w} + \overline{y} + \overline{z}) \cdot (w + y + \overline{z})$

3 Komplexität

```
1 [...]
2 for(i=0; i<n; i++){
3   [...]
```

```
4    if(k>0){
5       m=i;\\
6       While(m<n){
7        comp()
8  } } }
```

3.1 Bestimmen Sie die Komplexität der gegeben Methode im Best-Case. Was muss für k dabei erfüllt sein? Nehmen sie an, dass die Funktion comp() aus Zeile 7 einen Aufwand von T(n)=2 besitzt.

3.2 Bestimmen Sie den Worst-Case der Methode.

4 Traversierung von Graphen

4.1 Breitensuche Handsimulation

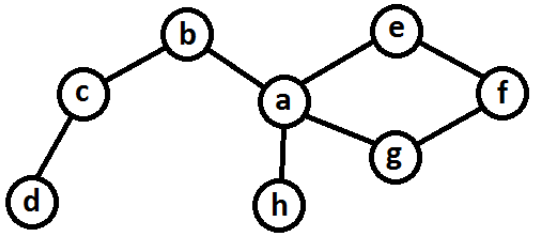
Traversieren Sie den Graphen G mit Breitensuche. Führen Sie dazu eine Handsimulation mit Hilfe der untenstehenden Tabelle durch. Dabei bezeichne t die Nummer des aktuellen Schleifendurchlaufs und EK den aktuellen zu expandierenden Knoten.

Befolgen Sie bei der Simulation folgende Regeln:

- Startknoten ist der Knoten mit Bezeichner a
- Geben Sie für $t = 0$ den Zustand des Stacks unmittelbar nach Initialisierung an.
- Geben Sie für $t > 0$ den Zustand des Stacks jeweils am Ende des aktuellen Schleifendurchlaufs an.
- Notieren Sie schrittweise den aktuellen Knoten und den aktuellen Inhalt der Queue, es soll immer nur ein Knoten pro Schritt in die Queue eingefügt werden

4.2 Weleches Suchverfahren fürht mit den wenigsten Schritten zum Ziel?

- Startknoten: a
- Gesucht: d



5 Allgemeine Fragen zur Implementierung

- Frage zur generischen Klassen Implementierung
- wofür wird compareTo() nicht benutzt?

- Wofür wird Iterierbarkeit nicht benutzt?
- ...

6 AVL-Baum

Gegeben ist einallgemeiner AVL-Baum.

6.1 Fügen Sie das Blatt (Blatt) in den gegeben AVL-Baum ein. Führen die dabei keine rotation durch.

6.2 Löschen Sie das Blatt (Blatt) aus dem ursprünglichen (nicht den aus Aufgabe 6.1) AVL-Baum.

6.3 Rotieren Sie den Baum aus Aufgabe 6.2 nach der Lösch-operation so, dass die AVL-Bedingung wieder erfüllt ist

7 Mergesort

Schreiben Sie für die gegebene Klasse die Methode 'private static void merge(int[] f, int[] g, int start, int end, int p)'

8 Stack

8.1 Klasse Stack implementieren

Implementrieren sie einen abstrakten Datentyp Stack$< T >$, der einen Stack mit Elementen eines generischen Grundtyps T realisiert. Die realisierung soll ausschließlich auf der doppelt verketteten Liste LinkedList$< T >$ aus der Java standard-Bibliothek basieren. Der Stack$< T >$ besitzt ein Attribut private LinkedList$< T >$ list und soll die folgenden Operatioinen bereitstellen:

1. Einfügen eines Elements
2. Entfernen eines Elements
3. Ausgeben eines Elements (peek)

Hinweis: Um LinkedList einzubinden können Sie den Befehl 'import java.util.LinkedList;' verwenden. Die Java Standard-Bibliothek stellt folgende Befehle zur verfügung:

- 'push()' fügt ein Element am Anfang der Liste ein
- 'pop()' entfernt das erste Element der Liste und gibt dieses zurück. Existiert kein Element, so wird die Null-Referenz zurückgegeben.
- 'top()' gibt das erste Element der Liste zurück.

...

8.2 Testklasse

Implementieren sie eine Testklasse zum Testen Ihrer Stackklasse.Dabei sollen der Reihe nach:

- 'Amerika', 'Afrika', 'Asien' in den Stack eingefügt werden
- Die Methode 'peek' benutzt werden und notiert werden, was dabei auf der Konsole ausgegeben wird
- drei mal 'pop' benutzt werden und anschließend notiert werden, was auf der Konsole ausgegeben wird.

9 LinkedList

Gegeben sei eine Klasse LinkedList einer einfach verketteten Liste.

9.1 Schreiben Sie eine Methode 'peek', die das erste Element der Liste wiedergibt.

9.2 Schreiben Sie die Methode 'public void addLast()'. Beachten Sie dabei die Struktur einfach verketteter Listen.

10 Binärer Suchbaum

Gegeben sei ein Binärer Suchbaum mit diversen Unterklassen (Fork, Leaf, ...).

10.1 Schreiben Sie eine Methode, die in der Unterklasse 'Leaf' ein Element sucht, indem sie mit einem bestimmten 'Key' vergleicht und den Dateninhalt zurückgibt.

gegeben war bereits der Aufruf der Such-Methode.

10.2 Schreiben Sie eine Methode, die in der Unterklasse 'Fork' ein übergebenes Element mit einem 'Fork-Knoten' vergleicht, sodass der Baum am richtigen 'Kind' des 'Fork-Knotens' weiter Traversiert werden kann.