



Übung 4

Aufgabe 4.1: Verwendung von Listen (5+2)

(7 Punkte)

Eine Menge (englisch Set) ist eine Datenstruktur, in der jedes Element nur einmal vorkommen darf (vgl. Menge in der Mathematik).

- a) Vervollständigen Sie in der im Ilias zur Verfügung gestellten Klasse ListSet die Methoden add, remove, union (Vereinigung), intersect (Schnitt) und subtract (Differenz).
- b) Modifizieren Sie Ihre Implementierungen aus a), sodass die union Methode bei einer Vereinigung von einer Menge mit n Elementen mit einer Menge mit m Elementen im schlechtesten Fall eine Laufzeit von $\mathcal{O}(n + m)$ besitzt.

Aufgabe 4.2: Adaptive Listen

(3 Punkte)

Erstellen Sie zwei Unterklassen der gegebenen Klasse SimpleLinkedList, welche die ersten beiden in der Vorlesung vorgestellten Strategien für adaptive Listen unterstützen (Move-To-Front, Transpose). Überschreiben Sie dazu die get(int)-Methode und reorganisieren Sie die Listenelemente entsprechend der Strategie.

Aufgabe 4.3: Stacks und Queues (3+2)

(5 Punkte)

- a) Gegeben sei ein Array der Größe n . Beschreiben Sie ein Verfahren, um 2 verschiedene Stacks innerhalb von diesem einem Array zu verwalten, solange die Summe der Größe der beiden Stacks kleiner gleich n ist. Geben Sie Pseudocode für die push und pop Operationen an, welche jeweils eine Laufzeitkomplexität von $\mathcal{O}(1)$ aufweisen. Fangen Sie den Fall, dass die Summe der beiden Stackgrößen n überschreitet, in Ihrem Pseudocode an entsprechender Stelle ab.
- b) Eine Queue ist eine Warteschlange nach dem FIFO-Prinzip (First In – First Out). Dabei fügt die Operation add ein Element in die Queue ein und die Operation poll entnimmt das Element, welches sich am längstem in der Queue befindet.

Führen Sie die folgenden Operationen manuell auf einer Queue aus. Welche Zahl wird durch die letzte Anweisung ausgegeben? Wie sieht die Queue nach der Ausgabe aus? Der Rechenweg soll ersichtlich sein.

```
1 add(5);
2 add(2);
3 poll();
4 add(13);
5 add(poll()*poll());
6 add(13);
7 add(3);
8 add(poll()+poll());
9 add(9)
10 add(poll()/poll());
11 System.out.println(poll());
```

Aufgabe 4.4: Skip-Listen*

(5 Punkte)

Stellen Sie für das Interface RandomSkipList eine *generische* Implementierung einer randomisierten Skip-Liste bereit.

*** Aufgabe 4.4 ist für Lehramtsstudierende optional.**