ALP 2, Hannes Federrath, Sommersemester 2001

Probeklausur ALP 2

Insgesamt sind 42 Punkte erreichbar. Zum Bestehen sind 50 % der Punkte nötig.

4 Aufgaben, Bearbeitungszeit: 180 Minuten

Name: .....

Vorname: .....

Matrikelnummer: .....

Tutor: .....

Aufgabe 1: (6 Punkte)

Gegeben sei eine einfach verkettete Liste `EList` von Objekten der Klasse `Node`.

```
class EList {
    Node first = null;
    void addNode(Node node) {
        node.next = first;
        first = node;
    }
    void reverse() {
        // zu implementieren!
    }
}
class Node {
    Node next = null;
    Object content;
    Node(Object content) { this.content = content; }
}
```

Implementiere innerhalb der Klasse `EList` eine Methode `reverse()` zum „Umdrehen“ der Reihenfolge der Listenelemente. Die Variable `next` des letzten Listenelementes enthält `null`.

Aufgabe 2: (10 Punkte)

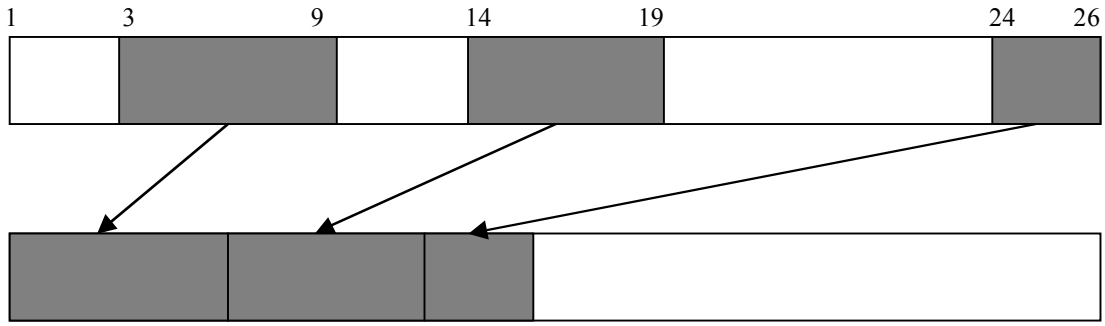
a) (5 P) Schreibe eine möglichst laufzeiteffiziente Funktion, die die Anzahl der Elemente eines Feldes `a` von Integer-Zahlen x ($0 \leq x \leq 99$) liefert, die nicht mehrfach in `a` vorkommen.

b) (5 P) Schreibe eine möglichst laufzeiteffiziente Funktion, die alle Duplikate aus a durch `-1` ersetzt. (Ist `a[i]==a[j]` mit `i!=j`, dann erhält `a[i]` oder `a[j]` den Wert `-1`.

Aufgabe 3: (16 Punkte)

Der Speicher eines Rechners enthalte zu einem bestimmten Zeitpunkt freie und belegte Bereiche in irgendeiner Reihenfolge. Wenn man einen großen Speicherbereich benötigt, müssen die belegten Bereiche „kompaktifiziert“ werden, um einen großen Bereich frei zu machen.

Beispiel:



Der Speicher sei definiert durch

```
final int N = ...; // im Beispiel oben ist N=27
byte[] speicher = new byte[N];
```

Die Adressen des Speichers werden durch die Indizes `0` bis `N` modelliert. Dabei fungieren `0` und `N` als Pseudoadressen: die Zelle `N` existiert nicht und die Zelle `0` bleibt grundsätzlich unbelegt. Dies erleichtert die Lösung der Aufgabe. Der tatsächliche Speicher kann von `1` bis `N-1` adressiert werden.

Die aktuelle Speicherbelegung mit höchstens `MAX` belegten Bereichen wird mit Hilfe zweier Felder festgehalten:

```
int[] anfang = new int[MAX+2];
int[] ende   = new int[MAX+2];
```

Die Einträge von `anfang[i]` und `ende[i]` mit i ≥ 1bezeichnen die Adressen des ersten und letzten Bytes im i-ten belegten Bereich. Als Endemarkierung gibt es hinter dem letzten gültigen Eintrag in `anfang` einen Eintrag mit der Pseudoadresse `N`. Außerdem enthält `ende[0]` stets die Pseudoadresse `0`.

Für die Beispiel oben gezeigte Belegung sehen `anfang` und `ende` so aus:

	anfang	ende
0		0
	3	9
	14	19
	24	26
	N	
max+1		

Implementiere eine Java-Prozedur `compactify()` zum kompaktifizieren, die die Daten im Speicher entsprechend verschiebt und die Verwaltungsdaten entsprechend anpasst.

Aufgabe 4: (10 Punkte)

Die Zahl π lässt sich mit Hilfe eines Kettenbruchs folgendermaßen darstellen:

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \ldots}}}$$

a) Implementiere ein rekursives Programm, das π mit Hilfe von *n* Termen des Kettenbruchs berechnet. Der *n*-te Term sei dabei definiert als 2+0.

b) Schreibe dafür ein iteratives Programm und bestimme dessen asymptotische Laufzeit.

Viel Erfolg!