Universität Mannheim
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Christoph Kuhmünch, Gerald Kühne

Praktische Informatik I
WS 1999/2000

Übungsblatt 6 **Ausgabe: Mi, 01.12.99** **Abgabe: Di, 07.12.99, 18 Uhr**

Aufgabe 1: Wiederholungsanweisungen [8 Punkte]

Im Folgenden beschäftigen wir uns mit einem einfachen Verschlüsselungsverfahren. Ein Block B mit N Zeichen $(b_0, b_1, \ldots, b_{N-1})$ soll verschlüsselt werden. Dazu werden die Zeichen des Blocks mittels eines vorgebenen Schlüssels $S = (s_0, s_1, \ldots, s_{N-1})$ permutiert. Ein einzelner Schlüsselwert $s_i \in \{0, 1, \ldots, N-1\}, s_i \neq s_j$ für $i \neq j$, gibt dabei an, auf welche Position das Zeichen b_i gesetzt werden soll.

Das folgende Beispiel soll das Verfahren verdeutlichen: Der Block $B = (H, a, l, l, o)$ mit $N = 5$ Zeichen wird mit Hilfe des Schlüssels $S = (4, 3, 2, 1, 0)$ verschlüsselt. D.h. Zeichen b_0 wird an Position 4 gesetzt, da $s_0 = 4$, Zeichen b_1 wird an Position $s_1 = 3$ gesetzt usw. Als Ergebnis erhalten wir schließlich $B' = (o, l, l, a, H)$.

(a) [5 Punkte] Implementieren Sie eine Verschlüsselungskomponente als Java-Application und geben Sie das Programm über abox ab. Benutzen Sie als Klassennamen `Enkodierung`. Das Programm muß die folgenden Kommandozeilenparameter erwarten: (1) Die zu verwendende Blockgröße N als Integerzahl, (2) Dateiname der Datei, die den Schlüssel S enthält (Format: durch Leerzeichen getrennte Integerzahlen), (3) Dateiname der zu verschlüsselnden Datei. Als Ergebnis muß die verschlüsselte Datei über die Standardausgabe ausgegeben werden.

Beachten Sie, daß eine zu verschlüsselnde Datei beliebig viele Blöcke enthalten kann. Desweiteren sind alle Textzeichen (d.h. auch Zeilenumbrüche) mit einzubeziehen. Läßt sich ein Block nicht vollständig mit Zeichen aus der Datei füllen (Dateilänge L mod Blockgröße $N \neq 0$), so verwenden Sie das Leerzeichen als Füllwert.

Hier ein Beispiel für einen Kommandozeilenaufruf und der Ergebnisausgabe des von Ihnen zu entwickelnden Programms:

```
schluessel.txt:   4 3 2 1 0
superwichtig.txt: Hallo
> java Enkodierung 5 schluessel.txt superwichtig.txt
ollaH
```

(b) [3 Punkte] Verweisen Sie auf verwendete Schleifenkonstrukte und begründen Sie (im Programmtext als Kommentar), weshalb Sie den jeweiligen Schleifen-Typ (`for, while, do { ... } while`) verwendet haben.

Aufgabe 2: Modularität [12 Punkte]

Entwickeln Sie einfache Module für die Verwaltung von endlichen (mathematischen) Mengen. Gehen Sie dabei davon aus, daß nur die Zahlen 0 bis 999 in den Mengen vorkommen. Beachten Sie dabei, daß in einer Menge jedes Element nur einmal vorkommen darf. Entwerfen Sie Funktionen für die folgenden Operationen:

1. Initialisieren einer Menge als leere Menge.

2. Prüfen, ob eine Menge leer ist.
3. Feststellen, wieviele Elemente in einer Menge enthalten sind.
4. Feststellen, ob ein bestimmtes Element x in einer Menge enthalten ist.
5. Ein Element x in eine Menge einfügen.
6. Die Schnittmenge zweier Mengen M_1 und M_2 bilden.
7. Die Vereinigungsmenge zweier Mengen M_1 und M_2 bilden.

Implementieren Sie nun unter dem Namen `Mengenverwaltung` eine Java-Application und geben Sie diese über abox ab. Das Programm muß als Kommandozeilenparameter zwei Dateinamen erwarten. Die entsprechenden Dateien enthalten endliche Mengen (Format: durch Leerzeichen getrennte Integerzahlen). Folgende Operationen muß das Programm nacheinander ausführen:

1. Einlesen der beiden Mengen M_1 und M_2.
2. Ausgabe der Anzahl der Elemente von M_1.
3. Ausgabe der Anzahl der Elemente von M_2.
4. Berechnung der Schnittmenge.
5. Ausgabe, ob die Schnittmenge leer ist (ja/nein).
6. Berechnung der Vereinigungsmenge.
7. Ausgabe der Anzahl der Elemente der Vereinigungsmenge.
8. Ausgabe, ob der Wert 42 in der Vereinigungsmenge enthalten ist (ja/nein).

Hier ein Beispiel für einen Kommandozeilenaufruf und der Ergebnisausgabe des von Ihnen zu entwickelnden Programms:

```
menge1.txt:  1 2 3 4 5
menge2.txt:  5 6 7 8 9
> java Mengenverwaltung menge1.txt menge2.txt
5
5
nein
9
nein
```