TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

WS 2007/2008
Aufgabenblatt 3
2. November 2007

Abgabe: Montag, 12.11.07, 8:00, per Mail/Papier beim jeweiligen Tutor

Praktikum Grundlagen der Programmierung

Aufgabe 14 (Ü) Kontrollfluss-Graph

Zeichnen Sie für das angegebene MiniJava-Programm den Kontrollfluss-Graphen.

```
int i, j, k, n;
i = read();
k = -1;
j = read();
n = 0;
while( j >= -k){
    if( i > j )
        i = i - j;
    else
        j = j % i;
    n = (n - k) * 2 - 1;
}
write(i);
```

Aufgabe 15 (Ü) Minimum und Maximum

Schreiben Sie ein Mini-Java-Programm namens `MinMax.java`, das in einem Array von ganzen Zahlen den kleinsten und den größten Wert findet. Das Programm soll sich wie folgt verhalten:

- Zunächst fragt das Programm ab, wie viele Zahlen in das Array eingegeben werden sollen.
- Dann werden die Zahlen eingegeben und in einem Array gespeichert.
- Anschließend wird das Array durchsucht und in **einem Durchgang** soll sowohl die kleinste als auch die größte Zahl gefunden werden.
- Schließlich sollen die kleinste und die größte Zahl ausgegeben werden.

Aufgabe 16 (Ü) Tanzpaarbildung

In dieser Aufgabe wollen wir aus einem Array von Männern und einem Array von Frauen Tanzpaare bilden. Um ein geeignetes Tanzpaar zu bilden, wird aus allen restlich verfügbaren Männern der grösste Mann und aus den restlich verfügbaren Frauen die grösste Frau ausgewählt. Schreiben Sie hierzu ein MiniJava-Programm, das aus den beiden Arrays ein zweidimensionales Array mit den Tanzgruppierungen erstellt und geben Sie die Tanzpaare am Bildschirm aus. Um die Tanzpartner einzulesen, steht Ihnen die Methode `readString()` der Klasse MiniJava zur Verfügung. Die Grösse eines Tanzpartners kann mit Hilfe der Methode `length()` ermittelt werden. Zum Beispiel:

```
String name = readString();
int groesse = name.length();
```

Aufgabe 17 (Ü) Sieb des Eratosthenes

Folgendes Verfahren zur Primzahlen-Gewinnung ist als **Sieb des Eratosthenes** bekannt:

Aus der Menge der natürlichen Zahlen von 2 bis n werden alle Nicht-Primzahlen gestrichen. Beginne mit der Zahl 2 und streiche alle echten Vielfachen von 2. Streiche nun sukzessiv alle echten Vielfachen der jeweils nächsthöheren noch nicht gestrichenen Zahl. Nach Streichung aller Vielfachen enthält die Restmenge nur noch Primzahlen.

Implementieren Sie das oben beschriebene Verfahren in Java mit Hilfe eines Arrays. Ihr Programm soll nach Festlegung einer Obergrenze durch den Benutzer alle Primzahlen bis zu dieser Grenze bestimmen und ausgeben.

Aufgabe 18 (H) Kontrollfluss-Graph (4 Punkte)

Zeichnen Sie für das angegebene MiniJava-Programm den Kontrollfluss-Graphen.

```
int n,sum,prod,i;
n = read();
if (n >= 0){
     sum = 0;
     prod = 1;
     i = 0;
     while (i != n) {
          sum = sum + prod;
          prod = 3 * prod;
          i = i + 1;
     }
     sum = 2 * sum + 1;
     write(sum);

} else {
     write("n<0");
}
```

Aufgabe 19 (H) Natürliche Zahlen (5 Punkte)

Um mit beliebig großen natürlichen Zahlen rechnen zu können, sollen in dieser Aufgabe natürliche Zahlen als Arrays von int-Werten aus dem Intervall $[0,9]$ in natürlicher Weise repräsentiert werden. Die Zahl 12345 wird also durch das Array $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}$ dargestellt.

a) Schreiben Sie ein MiniJava-Programm, das zwei Zahlen als String einliest, auf oben beschriebene Art repräsentiert und die Summe der eingelesenen Zahlen ausgibt.
b) Schreiben Sie ein MiniJava-Programm das zwei Zahlen als String einliest, auf oben beschriebene Art repräsentiert und das Produkt der eingelesenen Zahlen ausgibt.

Hinweis: Mit der Methode `readString()` der Klasse MiniJava können Sie Strings vom Benutzer eingeben lassen. Repräsentiert der String `zahl` eine natürliche Zahl so können Sie die `i`-te Ziffer mit dem Ausdruck `(int)(str.charAt(i-1)) - (int)'0'` erhalten.

Zum Beispiel:

```
String eingabe = "56789";
```

Um die Ziffer an Position 3 zu erhalten (also die Ziffer 7), ist folgender Aufruf nötig:

```
int ziffer = (int)(eingabe.charAt(2)) - (int)'0';
```

Aufgabe 20 (H) Fussballturnier **(6 Punkte)**

In dieser Aufgabe sollen Sie ein Fussballturniersystem implementieren. Die teilnehmenden 8 Mannschaften sind durch ein Array, das den Namen der Mannschaft speichert und ein zweites Array, welches die erzielten Punkte der Mannschaft fasst, dargestellt. Es sind insgesamt 3 Runden in diesem Turnier zu spielen, um den Sieger des Turniers zu ermitteln. In einer Spielrunde treten jeweils zwei Mannschaften gegeneinander an. Der Sieger eines Spiels ist eine Runde weiter, der Verlierer ist vom Turnier ausgeschieden, so dass in einer Runde gemäss des KO-Systems die Hälfte der Mannschaften rausfliegt.
Zuerst werden die Spielpaarungen in einer Turnierrunde ermittelt. Hierzu soll jeweils die Fussballmannschaft mit den niedrigsten Punkten gegen die Mannschaft mit den meisten Punkten spielen, die mit dem zweitniedrigsten Punktestand gegen die mit dem zweithöchsten Punktestand, usw.
Bei einem Spiel zweier Mannschaften wird das Ergebnis dieser Begegnung durch den Aufruf der Methode `int[] getScores()` der Klasse `Match` ermittelt. Die Methode liefert ein Array der Länge 2 mit den erzielten Toren der beiden Mannschaften. `getScores()` garantiert, dass ein Spiel so lange dauert bis eine Mannschaft mehr Tore als ihr Gegner erzielt hat, es also kein Unentschieden gibt. Bei Ihrer Implementierung des Fussballturniersystems sollen jeweils die Sieger einer Runde und der Gesamtsieger des Turniers ausgegeben werden.

Achten Sie auf einen geschickten Einsatz der Kontrollstrukturen, um nicht unnötig doppelten Code zu produzieren!