Technische Universität München
Fakultät für Informatik
Prof. Dr. Seidl

WS 2004/2005
Übungsblatt 4
18. November 2004

Übungen zu Einführung in die Informatik I

Aufgabe 13 Übersetzung von MiniJava nach MiniJVM

Übersetzen Sie das folgende MiniJava-Programm in ein MiniJVM-Programm **nach den Regeln** aus der Vorlesung. Die Variable `x` soll dabei in Speicherzelle `0`, Variable `y` in Speicherzelle `1`, Variable `z` in Speicherzelle `2` und Variable `res` in Speicherzelle `3` stehen.

```
x = 1;
y = 2;
z = 5*x - y/2;
if (z<y) res = z;
else res = y;
```

Regeln aus der Vorlesung zur Übersetzung von:

`int` $x_0,\ldots,x_{n-1}$`;`	=	ALLOC n
`x`	=	LOAD i — sofern `x` die *i*-te Variable ist.
`17`	=	CONST 17
`-` expr	=	Übersetzung von expr NEG
$expr_1$ + $expr_2$	=	Übersetzung von $expr_1$ Übersetzung von $expr_2$ ADD
`x =` expr`;`	=	Übersetzung von expr STORE i — sofern `x` die *i*-te Variable ist.
`x = read();`	=	READ STORE i — sofern `x` die *i*-te Variable ist.
`write(` expr`);`	=	Übersetzung von expr WRITE
`if (` cond `)` $stmt_1$ `else` $stmt_2$	=	Übersetzung von cond FJUMP A Übersetzung von $stmt_1$ JUMP B A: Übersetzung von $stmt_2$ B: ...
`while (` cond `)` $stmt_1$	=	A: Übersetzung von cond FJUMP B Übersetzung von $stmt_1$ JUMP A B: ...
$stmt_1$ ... $stmt_k$	=	Übersetzung von $stmt_1$ ... Übersetzung von $stmt_k$
prog	=	ALLOC n Übersetzung von ss HALT

Aufgabe 14 Quicksort

Ziel dieser Aufgabe ist es den Quicksort-Algorithmus über Reihungen ganzer Zahlen zu implementieren. Er hat folgende Ablaufstruktur:

- Wähle aus dem zu sortierenden Bereich ein beliebiges Element, das sogenannte Pivot-Element.
- Ordne die Inhalte des Arrays so um, dass links vom Pivot-Element nur Elemente sind, die kleiner oder gleich als das Pivot-Element (Links-Partitionierung) und rechts nur solche, die größer oder gleich als das Pivot-Element sind (Rechts-Partitionierung).
- Wende den Algorithmus auf die Links-Partitionierung und auf die Rechts-Partionierung an.
- Sobald die zu sortierenden Teilbereiche maximal die Länge 1 haben, bricht das Verfahren ab.

Zur Implementierung benötigen wir einige Hilfsfunktionen, die vorab programmiert werden sollen.

a) (Tausch von Feldelementen) Entwickeln Sie eine Funktion namens `swap`, die in einem Feld ganzer Zahlen, die Einträge mit den Indizes i und j vertauscht.

b) (Partitionierung eines Feldes) Gesucht ist eine Funktion der Signatur `int partitionIt(int[] reihung, int left, int right, int pivot)`. Diese Funktion soll die Elemente des Feldes `reihung` derart vertauschen, dass links des Elementes mit dem Wert `pivot` nur kleinere oder gleiche Zahlen und rechts davon nur größere oder gleiche Zahlen stehen. Zurückgegeben wird der Index des Pivotelementes.

c) Implementieren Sie mit Hilfe der bisher erarbeiteten Funktionen den Quicksortalgorithmus.

d) Erstellen sie eine `main`-Routine, in der der Quicksortalgorithmus für ein Feld, dessen Elemente zufällig mit Zahlen belegt sind, getestet wird. (**Hinweis:** Eine Zufallszahl zwischen 0 und $n-1$ läßt sich mit dem Befehl `int zahl = (int) (n*Math.random())` gewinnen.

Aufgabe 15 Mustererkennung

a) Schreiben Sie ein Programm, das eine aus `0` und `1` Symbolen bestehende Folge mit fester Länge zufällig generiert und darin die längste Teilfolge bestehend ausschließlich aus `0` Symbolen findet. (**Hinweis:** Eine binäre Zahl (0 oder 1) kann zufällig mit dem Ausdruck `(int)(Math.random()+0.5)` generiert werden)

b) (**Zusatzaufgabe**) Schreiben Sie ein Programm, das ein aus `0` und `1` Symbolen bestehendes zweidimensionales Array mit festen Dimensionen zufällig generiert und darin das größte Rechteck bestehend nur aus `0` Symbolen findet.