Informatik I WS 06/07
Prof. Dr. C. Damm
Dipl.-Inform. Marc Njoku

Übungsblatt 10

Ausgegeben am:	10.01.2007
Abgabe bis:	19.01.2007

Thema: Sortieralgorithmen, *O*-Notation, Java, Korrektheitsbeweis, ADT

Die Theorieaufgaben auf diesem Blatt sind bis zum Freitag, 19. Januar um 14.00 Uhr in die Info-I-Briefkästen im Erdgeschoss der NAM einzuwerfen. Die Kästen sind nach Übungsgruppen geordnet. Achten Sie bitte darauf dass Sie Ihren Zettel in den richtigen Kasten werfen, und dass **Name und Gruppe** auf **jedem Blatt** stehen! Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen. Die Praxis-Aufgaben bearbeiten Sie bitte am Rechner und führen Sie Ihrem Tutor am Rechner vor. Eine Bearbeitung in Zweier-Teams innerhalb Ihrer Übungsgruppe ist möglich.

Aufgabe 1 (Theorie: 24 Punkte):

Mergesort

1. Unten finden Sie die zwei Methoden `mergesort` und `vereinige`. Wählen Sie für beide Methode in jeder Zeile eine der drei Möglichkeiten aus und schreiben Sie den entsprechenden Buchstaben in die letzte Spalte der Tabelle.
   In jeder Zeile ist genau eine Möglichkeit richtig

A	B	C	Antwort
`void mergesort(int [] a,int l, int r) {`			-
`if(l>r) {`	`if(l<r) {`	`if(l==r) {`	
`int m=(r+l)/2;`	`int m=(r-l)*2;`	`int m=r*l-2;`	
`mergesort(a,l,r);`	`mergesort(a,l,m);`	`mergesort(a,r,l);`	
`mergesort(a,m-1,l);`	`mergesort(l,m,r);`	`mergesort(a,m+1,r);`	
`vereinige(a,l,m,r);`	`vereinige(a,l,r);`	`mergesort(a,l,m,r)`	
	`}}`	`}`	
`}`			-

A	B	C	Antwort
`void vereinige(int[] a,int l, int m, int r) {`			-
`char i,j;`	`int i,j;`	`double i,j;`	
`for(i=m+1;i>l;--i) {aux[i-1]=a[i-1];}`	`for(i=0;i<j;++i) {aux[i]=a[j];}`	`while(i<j) { aux[++i]=a[--j];}`	
`for(m=j;m<r;++m) {aux[r-m+j]=a[m+1];}`	`for(j=r;r<j;++r) {aux[r-m+j]=a[2*j];}`	`for(j=m;j<r;++j) {aux[r+m-j]=a[j+1];}`	
`for(int k=0;k<l;++k) {`	`for(int k=0;k<r;++k) {`	`for(int k=l;k<=r;++k) {`	
`if(i<j) {`	`if(aux[j] < aux[i]) {`	`if(j<i) {`	
`a[k]=aux[j];`	`aux[k]=a[j];`	`a[k]=a[j];`	

`--j;`	`++j;`	`--m;`	
`} end;`	`} else if(j==i) {`	`} else {`	
`aux[k]=a[i];`	`a[k]=aux[i];`	`a[i]=aux[k];`	
`++j;`	`++i;`	`++aux;`	
`}}`			–

2. Sortieren Sie die Folge: 85, 1, 7, 2, 83, 19, 14, 12 mit `mergesort`. Geben Sie dabei nach jedem Aufruf von `vereinige` die Folge und die Werte für linke und rechte Grenze (l bzw. r) des Teilfeldes in Tabellenform an.

Aufgabe 2 (Theorie: 11 Punkte):

Lösungsstrategien

In der Programmierung gibt es u.a. drei Schlagwörter, die für Lösungsstrategien von Problemen stehen: "Divide-and-Conquer", "Top-Down" und "Bottom-Up".

Erläutern sie kurz die Verfahren und ihre Zusammenhänge.

Aufgabe 3 (Theorie: 5 Punkte):

Stabilität von Mergesort

Wir nennen ein Sortierverfahren *stabil*, wenn zwei Elemente der Eingabefolge die gleich groß sind, in der Ausgabefolge in derselben Reihenfolge erscheinen wie in der Eingabefolge. Beispiel: Man sortiert Personen erst alphabetisch nach Nachnamen, dann nach Geburtsjahr. Bei einem stabilen Sortieralgorithmus sind die Personen innerhalb eines Jahrgangs immer noch nach Nachnamen geordnet. Bei einem nicht-stabilen Algorithmus ist das nicht notwendigerweise der Fall.

Ist das Sortierverfahren MergeSort *stabil*?

Begründen Sie ihre Antwort.

Aufgabe 4 (Theorie: 14 Punkte):

Ägyptische Multiplikation

1. Man beweise die Korrektheit der "ägyptischen Multiplikation": (http://www.muenster.org/mauritz/matheserver/teller/aegypten/zahl2.html)

```
int multipliziereAegyptisch(int x, int y, int p) {
        while ( x >= 1) {
                if (x%2 == 1)
                        p = p + y;
                x = x/2;
                y = 2*y;
        }
        return p;
}
```

1. Man zeige, dass die Anzahl der Bits in der Binärdarstellung (ohne führende Nullen) einer

natürlichen Zahl x gegeben ist durch:
$\lceil \log_2 x+1 \rceil$

sowie durch

$\lfloor \log_2 x \rfloor + 1$

Aufgabe 5 (Praktisch: 26 Punkte):

Binäre Suche

1. Implementieren Sie den Algorithmus für die binäre Suche (Skript, Kapitel *"ALGORITHMISCHE TECHNIKEN"*, Abschnitt 1.4) als While-Programm in Java.
2. Testen Sie Ihre Implementierung, indem Sie das Verfahren auf das Java-Feld

   {1, 3, 4, 7, 8, 10, 11, 12, 15, 16, 19, 23, 33, 35, 55}

   für alle Zahlen von 1 bis 55 anwenden. Geben Sie auch an, ob der Wert gefunden wird, und wenn ja, an welcher Position. Zusätzlich soll auch die Anzahl der Iterationsschritte protokolliert werden.
   Formatieren Sie die Ausgabe ähnlich wie unten vorgegeben:

```
Wert            Position              Anzahl Iterationen
0               Nicht enthalten       4
1               An Position 0         4
2               Nicht enthalten       4
3               An Position 1         3
:               :                     :
:               :                     :
```

Lassen Sie das Programm in ihrer Tutorengruppe testieren.

Aufgabe 6 (Praktisch: 20 Punkte):

Stack in Java

Ein Stack in Java ist eine Datenstruktur, die nach dem LIFO-Prinzip *(last-in-first-out)* arbeitet. Die Elemente werden am vorderen Ende der Liste eingefügt und von dort auch wieder entnommen. Das heißt, die zuletzt eingefügten Elemente werden zuerst entnommen und die zuerst eingefügten zuletzt.

Gegeben ist die Klasse `intStack` *, die einen Stack bereits implementiert.

Weiterhin existiert die Klasse `binaerziffernvorwaerts` *, welche eine eingegebene natürliche Zahl in einer Binärzahl umwandeln kann.

Implementieren Sie die Klasse `binaerziffernvorwaerts` neu, indem letztere von der Klasse `intStack` Gebrauch macht und somit eine explizite Rekursion (wie aus der Methode `binaerziffern(int n)` aus `binaerziffernvorwaerts` zu entnehmen ist) umgeht.

Lassen Sie Ihre umgeschrieben Klasse binaerziffernvorwaerts testieren.

*: http://user.informatik.uni-goettingen.de/~damm/Informatik-I/java/