Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Simulation und Graphik
Prof. Dr. Holger Theisel

Magdeburg, 07.02.2011

Klausur „Einführung in die Informatik“

Matrikelnummer:		
Name:	Vorname:	Studiengang:
Blattanzahl:		
Unterschrift Student/in:	Unterschrift Aufsicht:	

bitte **nicht** ausfüllen:

Aufg. 1: ADT	**Aufg. 2:** Einfach verkettete Liste	**Aufg. 3:** Partielle Korrektheit	**Aufg. 4:** Komplexität in O-Notation	**Aufg. 5:** Register-maschine	**Aufg. 6:** Java Klassen-hierarchie
(von 6)	(von 7)	(von 5)	(von 3)	(von 3)	(von 6)

Aufg. 7: Sortier-algorithmen	**Aufg. 8:** Stabile Sortierung	**Aufg. 9:** Rekursion / Stack	**Aufg. 10:** Aufwand binäre Suche	**Aufg. 11:** Wahr oder falsch?	**Aufg. 12:** Bonus-Aufgabe	**Summe**
(von 4)	(von 3)	(von 4)	(von 4)	(von 5)	(von 2 Zusatzpkt.)	(von 50)

Hinweise für die Klausurbearbeitung

- Überprüfen Sie die Klausur auf Vollständigkeit (13 Blätter). Füllen Sie das Klausurdeckblatt gewissenhaft mit Namen, Vornamen und Studiengang aus. Schreiben Sie auf jedes Blatt Ihren Namen.
- Legen Sie alle für die Klausur benötigten Dinge, insbesondere Lichtbildausweis (Studentenausweis zur Anwesenheitskontrolle), Stifte, Verpflegung auf Ihren Tisch. Schalten Sie Ihre Handys aus.
- Verwenden Sie für Ihre Antworten den freien Platz nach den Aufgaben und ggf. die Rückseiten der Blätter. Melden Sie sich, wenn Sie zusätzliche leere Blätter benötigen.
- Benutzung roter / grüner Stiftfarbe und Bleistift ist untersagt. Geben Sie bei Rechenaufgaben vor jedem Schritt an, was Sie gerade ausrechnen. Bewertet wird der Rechenvorgang, das Endergebnis alleine reicht nicht.
- Die Benutzung unerlaubter Hilfsmittel (Taschenrechner, Bücher, Folien zur Vorlesung, Mobiltelefone) gilt als Täuschungsversuch und führt zu einer Bewertung der Prüfung mit „nicht ausreichend“ (5,0).
- Schreiben Sie deutlich! Unleserliche Passagen können nicht korrigiert werden.
- Beschränken Sie sich auf die geforderten Angaben und halten Sie Ihre Antworten kurz und präzise. Nicht geforderte Angaben ergeben keine zusätzlichen Punkte.

Viel Erfolg!

Name:_____

Aufgabe 1: ADT (6 Punkte)

Eine rationale Zahl (Bruch) wird durch die Angabe des Zählers (numerator) und des Nenners (denominator) repräsentiert.

Geben Sie die **Signaturen und Axiome** für den abstrakten Datentyp `RatNumber` für folgende Operationen an:

`mkRat:`	`Int x Int → RatNumber` Konstruktor, der aus zwei Zahlen vom Typ *Int* als Zähler und Nenner eine rationale Zahl konstruiert (ohne Vorbedingung, d.h. Nenner 0 wird nicht abgefangen)
`num:`	Selektor zum Zugriff auf den Zähler einer rationalen Zahl
`denom:`	Selektor zum Zugriff auf den Nenner einer rationalen Zahl
`add:`	Funktion zum Addieren von zwei rationalen Zahlen
`normalize:`	Eine Funktion zum Kürzen einer rationalen Zahl

Hinweis:
Sie können die Methode

`gcd:` `Int x Int → Int`

zur Berechnung des größten gemeinsamen Teilers von *x* und *y* als bekannt voraussetzen und nutzen.

Name:_____

Aufgabe 2: Einfach verkettete Liste (7 Punkte)

Schreiben Sie eine Methode

```
public static  SimpleList<Integer> merge
              (SimpleList<Integer> a, SimpleList<Integer> b)
```

die zwei bereits aufsteigend geordnete Listen vom Typ `SimpleList<Integer>` so zu einer neuen Liste vereinigt, dass in der Ergebnisliste alle Elemente in sortierter Ordnung vorliegen.

Hinweis:
Die Klasse `SimpleList<T>` steht mit den folgenden Methoden zur Verfügung:

```
public SimpleList();
public boolean isEmpty();
public int size();
public void addFirst(T o);
public T getFirst();
public T removeFirst();
public void addLast(T o);
```

Name:_____

Aufgabe 3: Partielle Korrektheit (5 Punkte)

Gegeben sei der folgende Algorithmus:

```
Algorithm    fac
Input:       n
Output:      x     ( = n!)
    /*    Vorbedingung: n>=0          */
i := n
x := 1
while (i>0)
do
    /*    Invariante: 0 ≤ i ≤ n ∧ x = n!/i!      */
    x := x*i
    i := i-1
od
return x
    /*    Nachbedingung:    x = n!    */
```

Beweisen Sie die partielle Korrektheit dieses Programms.

Name:_____

Aufgabe 4: Komplexität in O-Notation (3 Punkte)

Zu welchen Komplexitätsklassen in O-Notation in Abhängigkeit von n gehören folgende drei Anweisungsfolgen? Begründen Sie Ihre Antwort.

a)
```
for(int i=1; i<n; ++i)
      for(int j=1; j>0; --j);
```

b)
```
for(int i=1; i<n; i*=2);
```

c)
```
for(int i=1; i<n; ++i)
      for(int j=1; j<n*n; ++j);
```

Name:_____

Aufgabe 5: Registermaschine (3 Punkte)

a) Welche Funktion berechnet das folgende Programm einer **Registermaschine**

```
 1  CLOAD 1
 2  STORE 2
 3  LOAD 1
 4  IF c0 = 0 GOTO 12
 5  LOAD 2
 6  CMULT 2
 7  STORE 2
 8  LOAD 1
 9  CSUB 1
10  STORE 1
11  GOTO 4
12  END
```

bei einer gegebenen Konfiguration $b=1,\ c_0=0,\ c_1=x,\ c_2=0,\ldots$ im Register c_2 ?

Hinweise
Die Befehle haben folgende Bedeutung:

LOAD i,	$i \in \mathbf{N}_+$	$b' = b+1$	$c'_0 = c_i$	$c'_j = c_j$ für $j \geq 0$
CLOAD i,	$i \in \mathbf{N}$	$b' = b+1$	$c'_0 = i$	$c'_j = c_j$ für $j \geq 0$
STORE i,	$i \in \mathbf{N}_+$	$b' = b+1$	$c'_i = c_0$	$c'_j = c_j$ für $j \neq i$
CSUB i,	$i \in \mathbf{N}_+$	$b' = b+1$	$c'_0 = \text{if } (c_0 \geq i) \text{ then } c_0 - i \text{ else } 0$	
CMULT i,	$i \in \mathbf{N}_+$	$b' = b+1$	$c'_0 = c_0 \cdot i$	
GOTO i,	$i \in \mathbf{N}_+$	$b' = i$		
IF c0=0 GOTO i,	$i \in \mathbf{N}_+$		$b' = \text{if } c_0 = 0 \text{ then } i \text{ else } b+1$	
END		$b' = b$		

b) Zu welcher Komplexitätsklasse in O-Notation gehört dieser Algorithmus zur Berechnung dieser Funktion?

Name:_____

Aufgabe 6: Java Klassenhierarchie (6 Punkte)

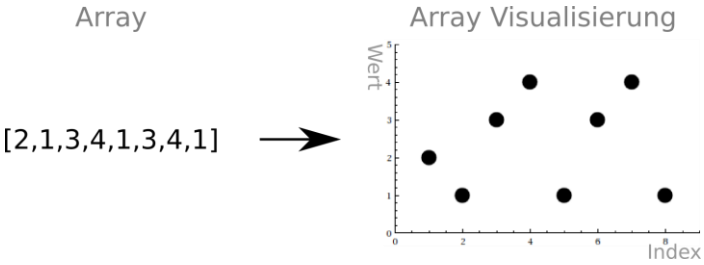
Gegeben ist folgendes Java Programm:

```
abstract class A {
  public String msg1()  { return "A"; }
  abstract public String msg2();
  private String msg3() { return "A"; }
}

class B extends A {
  public String msg2() { return "B"; }
  public String msg3() { return "B"; }
}

class C extends B {
  public String msg1() { return "C"; }
  public String msg2() { return "C"; }
  public String msg3() { return "C"; }
}

public class Hierarchy {
  public static void main(String[] args) {
    A a1 = new B();
    A a2 = new C();
    B b1 = new B();
    B b2 = new C();
    C c1 = new C();
    A a3 = c1;
    a2 = a1;
    System.out.print(a1.msg1());
    System.out.print(a1.msg2());
    System.out.print(a1.msg3());
    System.out.print(a2.msg1());
    System.out.print(a2.msg2());
    System.out.print(a2.msg3());
    System.out.print(a3.msg1());
    System.out.print(a3.msg2());
    System.out.print(a3.msg3());
    System.out.print(b1.msg1());
    System.out.print(b1.msg2());
    System.out.print(b1.msg3());
    System.out.print(b2.msg1());
    System.out.print(b2.msg2());
    System.out.print(b2.msg3());
    System.out.print(c1.msg1());
    System.out.print(c1.msg2());
    System.out.print(c1.msg3());
  }
}
```

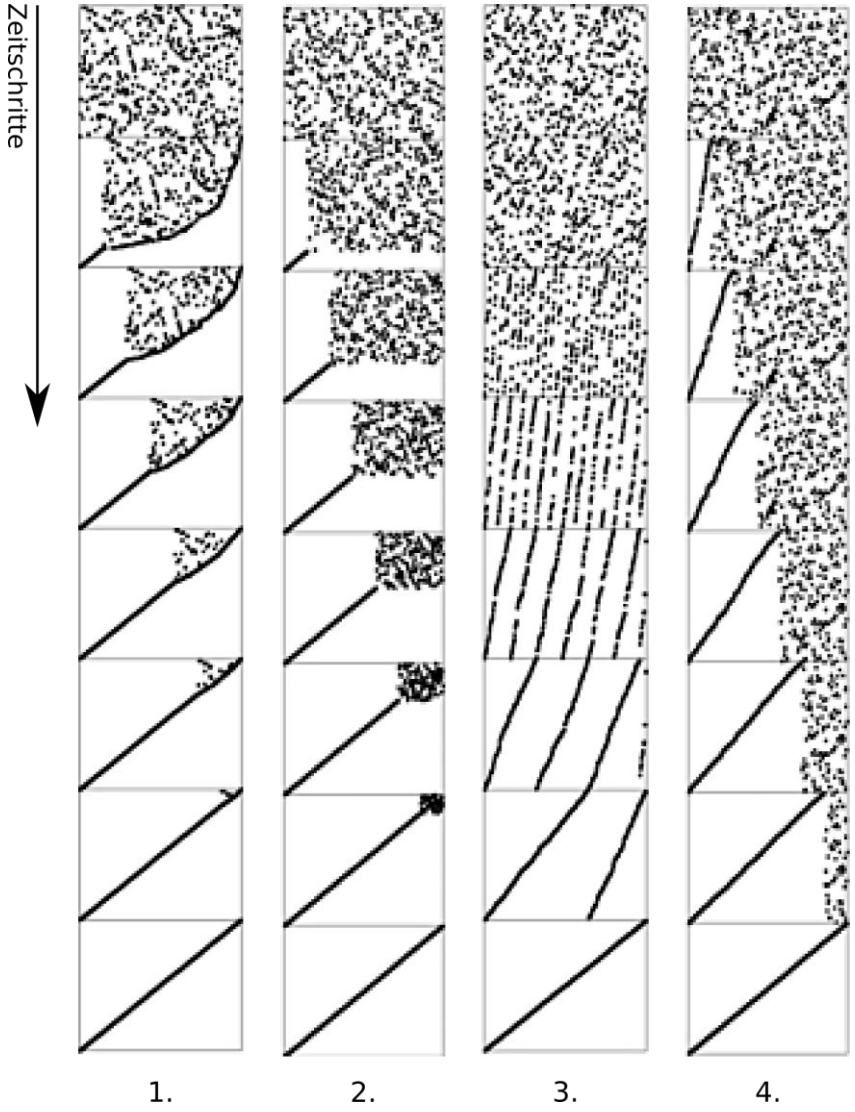

Was ist die Ausgabe dieses Programms? Beachten Sie, dass der angegebene Code Fehler enthalten kann (*compiler errors*). Kennzeichnen Sie diese gegebenenfalls und ignorieren Sie die entsprechenden Konstrukte für die Ausgabe.

Name:_____

Aufgabe 7: Sortieralgorithmen (4 Punkte)
Ein Array von Zahlen {1,2,…,n} kann als ein 2D-Feld visualisiert werden, indem auf der horizontalen Achse der Index und auf der vertikalen Achse der Wert über dem jeweiligen Index abgetragen wird (siehe Abbildung).



In dem folgenden Bild wird dies genutzt, um die Sortierung von zufälligen Folgen zu visualisieren.



Ordnen Sie jeder der vier Visualisierungen den entsprechenden Sortieralgorithmus zu, den Sie in der Vorlesung kennengelernt haben! Wie ist die Komplexität in O-Notation für das jeweilige Sortierverfahren im durchschnittlichen Fall?

Name:_____

Aufgabe 8: Stabile Sortierung (3 Punkte)

a) Wodurch unterscheidet sich ein stabiler Sortieralgorithmus von einem, der nicht stabil ist?

b) Geben Sie einen Algorithmus im Pseudocode an, der Arrays der Länge **3** stabil sortiert.

Hinweis:
Sie können eine Funktion `swap(a,b)` benutzen, die die Elemente `a` und `b` innerhalb eines Arrays vertauscht.

Name:_____

Aufgabe 9: Rekursion / Stack (4 Punkte)

Gegeben sei die rekursive Funktion

```
public static String what(String str){
     if (str.length()==0) return "";
     return what(str.substring(1)) + str.charAt(0);
}
```

a) Was macht `what`?

b) Schreiben Sie eine nichtrekursive Funktion

`public static String whatNew(String str),`

die einen *`Stack`* mit den üblichen Operationen *`push`* und *`pop`* nutzt und die gleiche Funktionalität wie *`what`* bietet.

Name:_____

Aufgabe 10: Aufwand binäre Suche (4 Punkte)

Leiten Sie einen Ausdruck zur Berechnung der im Mittel benötigten Anzahl an Vergleichen für das Suchen eines Elements in einem sortierten Array der Länge n mit dem Algorithmus der binären Suche her.

Hinweis:
Sie dürfen annehmen, dass n eine Zweierpotenz ist.

Name:_____

Aufgabe 11: Wahr oder falsch? (5 Punkte, wenigstens 0 Punkte)

Kreuzen Sie jeweils (ohne Begründung!) an, ob die Aussage wahr oder falsch ist. Für jede richtige Antwort gibt es einen halben Punkt, für jede falsche Antwort wird ein halber Punkt abgezogen. Beantworten Sie also lieber nur Fragen, bei denen Sie sich sicher sind!

	ja	nein
1. Die Ackermann-Funktion ergibt für jede positive Eingabe stets 91.		
2. Der `InsertionSort` Algorithmus kann je nach Einfügestrategie stabil und nicht stabil implementiert werden.		
3. Das Problem der Türme von Hanoi lässt sich in quadratischer Zeit in Abhängigkeit der Anzahl der zu verschiebenden Scheiben lösen.		
4. *Greedy*-Algorithmen finden stets das globale Optimum eines Problems.		
5. *Backtracking* findet stets ein lokales Optimum eines Problems.		
6. Die von einem deduktiven Algorithmus im Allgemeinen genutzte Schlussregel „*modus ponens*“ lässt sich mit $(A \wedge (A \rightarrow B)) \rightarrow B$ beschreiben, wobei A und B Aussagen sind.		
7. Es gibt unendlich viele nicht berechenbare Funktionen.		
8. Der reguläre Ausdruck `"([0-9]*[.][0-9]*)"` matcht nur ganze Zahlen.		
9. Das Halteproblem ist entscheidbar.		
10. Die Klasse der imperativen Algorithmen ist genau so mächtig wie die Klasse der funktionalen Algorithmen.		

Name:_____

Zusätzliche Bonus-Aufgabe (2 Punkte)

a) Wie viele Vergleiche benötigt man **mindestens**, um 4 verschiedene Zahlen zu sortieren? Begründen Sie Ihre Antwort.

b) Wie viele verschiedene Reihenfolgen von 4 Zahlen gibt es? Welchen Zusammenhang gibt es zwischen dieser Anzahl und dem Ergebnis von Aufgabenteil a)?