Name: Matrikelnummer:

Bachelorprüfung: Programmiermethoden und Techniken

SS15

Erlaubte Hilfsmittel: keine
Jede Griff zu einem elektronischen Gerät (z.B. Smartphone) wird als Täuschungsversuch gewertet.

Lösung ist auf den Klausurbögen anzufertigen. (eventuell Rückseiten nehmen)

Bitte legen Sie den Studentenausweis auf den Tisch.

Wird die Heftung der Klausur gelöst, ist auf jedem Blatt der Name einzutragen, ansonsten reicht der Name auf dem Deckblatt.

Bearbeitungszeit: 90 Minuten

Unterschrift

Benotung

Aufgabe:	1	2	3	4	5		Gesamt	Note
Punkte:	22	20	22	16	20		100	
erreicht:								

Aufgabe 1 (22 Punkte)
(Iteratoren in Java)

In dieser Aufgabe sollen Sie zwei Klassen schreiben, die die Schnittstelle `Iterator` implementieren.

a) (8 Punkte)

Schreiben Sie eine Klasse `CountDown`. Die Klasse soll die Schnittstelle `Iterator<Integer>` implementieren.

Die Klasse soll im Konstruktor eine positive Zahl erhalten.

Bei der Methode `next` soll die im Konstruktor übergebene Zahl zurückgegeben werden und für den nächsten Aufruf um eins verringert werden.

Der Iterator endet die Iteration, wenn die 0 erreicht ist.

Beispiel: Das Objekt `new CountDown(5)` iteriert über die Folge $5, 4, 3, 2, 1, 0$.

b) (8 Punkte)
Gegeben sei folgende Schnittstelle:

```
1 class Function<A>{
2   A apply(A a);
3 }
```

Listing 1: Function

Schreiben eine Klasse `AppyIterator<A>`, die `Iterator<A>` implementiert.

Sie soll zwei Objekte im Konstruktor übergeben bekommen: ein Objekt `f` des Interfaces `Function<A>` und ein Objekt `a` des generischen Typs `A`.

Ein Objekt der Klasse `ApplyIterator<A>` soll beim ersten Aufruf von `next()` das Objekt `a` zurückgeben. Beim zweiten Aufruf `f.apply(a)`. Bei jedem weiteren Aufruf soll das Ergebnis der Funktionsanwendung der Funktion `f` auf das Ergebnis der vorherigen Iteration zurück gegeben werden.

Es soll also über folgende Reihe iteriert werden:
$a, f(a), f(f(a)), f(f(f(a))), f(f(f(f(a)))), \ldots$

c) (3 Punkte)
Schreiben Sie eine Klasse `CountIterator`, die Unterklasse von `ApplyIterator<Integer>` ist.

Sie soll für eine Startzahl die Zahlen hoch zählen. Also `new CountIterator(42)` ergibt die Folge:
$42, 43, 44, 45, \ldots$.

Verwenden Sie in der Lösung einen Lambda-Ausdruck.

d) (3 Punkte)
Schreiben Sie eine Klasse `Repeat<A>`, die Unterklasse von `ApplyIterator<A>` ist.

Sie soll ein im Konstruktor übergebenes Objekt unendlich oft wiederholen. Also `new Repeat<>("Hossa")` ergibt die Folge:
`"Hossa"`, `"Hossa"`, `"Hossa"`, `"Hossa"`...

Verwenden Sie in der Lösung einen Lambda-Ausdruck.

Aufgabe 2 (20 Punkte)
(XML)

a) (8 Punkte)
Schreiben Sie mit dem DOM API (Paket `org.w3c.dom`) folgende Methode:
`public static void collectTagNames(Node node, Set<String> result);`
Sie soll alle Tagnamen von Elementknoten des übergebenen XML-Baums in die Ergebnismenge einfügen.

b) (8 Punkte)
Gegeben sei folgendes Interface

```
1 interface Property<A>{
2   boolean test(A o);
3 }
```

Listing 2: Property.java

Schreiben Sie mit dem DOM API folgende Methode:

`static void pathTo(Node n, Property<Node> p, List<Node> result)`
Es sollen in der Ergebnisliste `result` die Knoten eingefügt werden, die auf dem Pfad von der übergebenen Wurzel bis zum ersten Knoten mit positiven Test durch das `Property`-Objekt liegen.

Hat kein Knoten die gewünschte Eigenschaft, bleibt die Liste also leer.

c) (4 Punkte)
Schreiben Sie eine Methode:
`static void pathTo(Node n, String name, List<Node> result)`
Sie soll alle Knoten von der Wurzel zum ersten Knoten mit dem Knotennamen `name` in die Ergebnisliste einfügen.

Benutzen Sie bei der Implementierung einen Lambda-Ausdruck und die Methode `pathTo` aus Teil a).

Aufgabe 3 (22 Punkte)
(Strings in C)
Gegeben sei die folgende Struktur, die String-Objekte realisiert:

```
1 typedef struct{
2   unsigned int length;
3   char* data;
4 } String;
```

Listing 3: String.c

Schreiben Sie folgende Funktionen in C ohne Verwendung der Funktionen aus der Standardbibliothek `string.h`:

a) (6 Punkte)
`String reverse(String this)`

Es soll ein neuer String erzeugt werden, der aus den Zeichen des übergebenen Strings in umgekehrter Reihenfolge entsteht.

b) (6 Punkte)
`String append(String this, String that)`

Es soll ein neuer String erzeugt werden, der durch Aneinanderhängen der Zeichen der beiden übergebenen Strings entsteht.

c) (6 Punkte)
`unsigned int countWithProperty(String this,bool p(char))`
Es soll gezählt werden, für wieviel Zeichen im String die übergebene Funktion true ergibt.

d) (4 Punkte)
`unsigned int howManyCapitalLetters(String this)`

Es soll gezählt werden, wie viel Großbuchstaben in dem String enthalten sind. Benutzen Sie für Ihre Lösung einen Aufruf der Funktion `countWithProperty`.

Aufgabe 4 (16 Punkte)

Gegeben sei folgende Struktur, die eine Liste von ganzen Zahlen ausdrückt.

```
1 typedef struct {
2   int* array;
3   unsigned int capacity;
4   unsigned int size;
5 } AL;
```

Listing 4: AL.h

Das Attribut `array` dient dabei als Datenspeicher und hat die maximale Kapazität, die im Attribute `capacity` gespeichert ist. Das Attribute `size` gibt an, wie viele Elemente in der Liste gespeichert sind.

Folgende Funktionen zum Erzeugen und Löschen der Listenobjekte, sowie zum Hinzufügen eines weiteren Listenelements sind bereits implementiert:

```
6 AL* newAL();
7 void deleteAL(AL* this);
8 void add(AL* this, int elem)
```

Listing 5: AL.h

Schreiben Sie weitere Funktionen für diese Bibliothek.

a) (6 Punkte)
`bool isSorted(AL* this);`
Sie soll genau dann true ergeben, wenn die Elemente der Liste aufsteigend sortiert sind.

b) (6 Punkte)

`AL* filter(AL* list,bool p(int));`

Es wird ein neues Listenobjekt erzeugt, das aus all den Elementen der übergebenen Liste entsteht, für die die übergebene Funktion true ergibt.

c) (4 Punkte)

`AL* getPositiveNumbers(AL* list);`

Es wird ein neues Listenobjekt erzeugt, deren Elemente aus allen Zahlen größer 0 der übergebenen Liste bestehen. Benutzen sie zur Lösung die Funktion `filter`

Aufgabe 5 (20 Punkte)

Erklären Sie in kurzen Worten.

a) C Header Dateien beginnen oft mit einem `#ifndef`. Wozu dient dieses?

b) C kennt keine generischen Typen, wie sie aus Java bekannt sind. Wie kann man sich behelfen, wenn man trotzdem in C eine möglichst generische Datenstruktur entwickeln möchte, in der unterschiedliche Typen gespeichert werden können.

c) Die Klassen des neuen Date/Time APIs aus Java 8 haben alle keine öffentlichen Konstruktoren. Wie können Objekte dieser Klassen erzeugt werden?

d) Welche Zeichen sind in XML reservierte Symbole. Wie können Sie diese Zeichen trotzdem als Teil des Dokumententextes verwenden?

e) In Java 8 wurden zu dem Konzept der Iteratoren die Streams neu eingeführt. Welchen Vorteil bieten Streams? Nennen Sie auch ein paar typische Methoden von Streams.