Name: Matrikelnummer:

Bachelorprüfung: Programmiermethoden und Techniken

SS 20

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	6		Gesamt	Note
Punkte:	20	12	10	20	18	20		100	
erreicht:									

Aufgabe 1 (Iteratoren in Java)

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

a) (6 Punkte)

Schreiben Sie eine generische Klasse `Cycle<A>`. Die Klasse soll die Schnittstelle `Iterator<A>` implementieren.

Die Klasse soll im Konstruktor ein Objekt des Typs `java.util.List<A>` erhalten.

Die Methode `next` soll nacheinander die über Elemente der übergebenen Liste iterieren. Wurde über das letzte Listenelement iteriert, so soll wieder von vorne in der Liste begonnen werden. Es soll also potentiell unendlich immer wieder über die Elemente der Liste im Kreis iteriert werden.

b) (6 Punkte)

Gegeben sei folgende Record-Klasse:

```
1 record Pair<A,B>(A fst, B snd){}
```

Listing 1: Pair

Schreiben eine Klasse `Zip<A,B>`, die `Iterator<Pair<A,B>>` implementiert.

Sie soll zwei Objekte im Konstruktor übergeben bekommen: ein Objekt `itA` des generischen Typs `Iterator<A>` und `itB` des generischen Typs `Iterator<B>`.

Ein Objekt der Klasse `Zip<A,B>` soll beim Aufruf von `next` jeweils von beiden übergebenen Iteratoren `itA` und `itB` das nächste `A`- bzw. `B`-Objekt erfragen und ein `Pair<A,B>`-Objekt errechnen, welches das Ergebnis des Aufrufs ist.

Die Methode `hasNext` gibt solange `true` zurück, wie beide übergebenen Iteratoren noch weitere Elemente zurückgeben.

c) (4 Punkte)

Gegeben sei folgende Schnittstelle:

```
2 interface Zipper<A,B,C>{
3   C zip(Pair<A,B> p);
4 }
```

Listing 2: Pair

Schreiben Sie eine Klasse `Combine<A,B,C>` die `Iterator<C>` implementiert.

Sie soll zwei Objekte im Konstruktor übergeben bekommen: ein Objekt `itP` des generischen Typs `Iterator<Pair<A,B>>` und ein Objekt `f` der Schnittstelle `Zipper<A,B,C>`.

Beim Aufruf von `next` soll aus `itP` das nächste Paarobjekt genommen werden und mit der Funktion `f` das Ergebnis berechnet werden.

d) (4 Punkte)

Gegeben sei ein Objekt des Typs `Iterator<Pair<Integer,Integer>>`. Erzeugen Sie mit `Combine` ein Iteratorobjekt, das über die Produkte der beiden Zahlen der Paarobjekte iteriert.

Aufgabe 2 (Spliteratoren)

Gegeben sei die Klasse für eine Array-basierte Liste aus Aufgabe 1.

Die Klasse enthalte jetzt zusätzlich folgende Spliterator-Implementierung:

```
1   public Stream<A> stream(){
2     return StreamSupport.stream(new LiSplit(this,0,size),true);
3   }
4   private  class LiSplit implements Spliterator<A>{
5     Li<A> list;
6     int from;
7     int to;
8     LiSplit(Li<A> l,int f,int t){list = l;from=f;to=t;}
9
10    public boolean tryAdvance(Consumer<? super A> c){
11      if (to<=from) return false;
12
13      c.accept(list.data[from++]);
14      return true;
15    }
16
17    public int characteristics(){return 0;}
18    public long estimateSize(){return Long.MAX_VALUE;}
19
20    public Spliterator<A> trySplit(){
21      return null;
22    }
23  }
24}
```

Listing 3: Li.java

a) (8 Punkte)

Schreiben Sie die Methode `trySplit` der inneren Klasse `LiSplit` neu, so dass tatsächlich nicht `null` sondern ein neuer Spliterator zurück gegeben wird, der über die Hälfte der Liste iteriert und der Originalspliterator über die andere Hälfte.

Aufgabe 3 (XML)

a) (6 Punkte)

Gegeben sei folgende Schnittstelle:

```
1 interface Property{
2   boolean test(Node n);
3 }
```

Listing 4: Property

Schreiben Sie mit dem DOM API (Paket `org.w3c.dom`) die Methode `collectNodes`.

Sie soll den ganzen Baum durchlaufen und all die Baumknoten der Liste `java.util.List<Node> result` zufügen, für die das Schnittstellenobjekt `true` ergibt.

```
static void collectNodes(Node node, Property p,List<Node> result){









}
```

b) (4 Punkte)

Verwenden Sie die in a) entwickelte Methode, um alle Textknoten eines Baumes in die Liste result einzufügen.

```
static void collectTextNodes(Node node,List<Node> result){




}
```

Aufgabe 4 (Strings in C)

Gegeben sei die folgende Struktur, die String-Objekte realisiert:

```c
#include <stdbool.h>

typedef struct{
  unsigned int length;
  char* data;
} String;
```

Listing 5: String.c

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

a) (5 Punkte)

Schreiben Sie die Funktion `numberOfOccurrence(String this,char c)`.

Es soll berechnet werden, wie oft das Zeichen c im String enthalten ist.

```
int numberOfOccurrence(String this,char c){




}
```

b) (5 Punkte)

Schreiben Sie die Funktion `map(String this,char f(char))`.

Sie soll den String `this` so modifizieren, dass jeder Buchstabe durch die Funktion `f` verändert wird.

```
void map(String this,char f(char)){




}
```

c) (5 Punkte)

Schreiben Sie die Funktion `removeFirst(String* this)`.

Der String soll so modifiziert werden, dass das erste Zeichen gelöscht wird und der String somit ein Zeichen weniger enthält.

```
void removeFirst(String* this){




}
```

d) (5 Punkte)

Schreiben Sie die Funktion `reverse(String this)`.

Die Zeichen des Strings sollen so getauscht werden, dass sie in umgekehrter Reihenfolge erscheinen.

```
void reverse(String this){




}
```

Aufgabe 5 Gegeben sei folgende Datenstruktur für einen Binärbaum.

```
typedef struct BT{
  struct BT* left;
  int element;
  struct BT* right;
} BinTree;
```

Listing 6: Expression.h

Die linken und rechten Kinder können auch NULL sein. Wenn beide NULL sind, liegt ein Blatt vor.

Schreiben Sie folgende Funktionen für diese Datenstruktur.

a) (6 Punkte)

Schreiben Sie eine Konstruktorfunktion zum Erzeugen eines Baumes:

```
BinTree* newBinTree(BinTree* left, int e,BinTree* right){




}
```

b) (6 Punkte)

Schreiben Sie eine Methode, die testet, ob eine bestimmte Zahl `i` im Baum enthalten ist. (Es handelt sich nicht um einen binären Suchbaum.)

```
bool contains(BinTree* this, int i){

}
```

c) (6 Punkte)

Schreiben Sie eine Funktion, die einen Baum mit derselben Struktur erzeugt, dessen Knoten Zahlen haben, die mit der Funktion `f` aus der ursprünglichen Zahl an entsprechender Stelle berechnet wird.

```
BinTree* map(BinTree* this, int f(int)) {

}
```

Aufgabe 6 Erklären Sie in kurzen Worten.

a) Welche drei Phasen durchläuft ein Stream-Objekt in Java. Nennen Sie für jede Phase exemplarisch eine Methode.

b) In Java gibt es kein direktes Konstrukt, das die Aufgabe von `union` in C zur Bildung von Summentypen übernimmt. Wie kann man stattdessen die Summe zweier Typen realisieren.

c) Was passiert mit einem Iteratorobjekt, nachdem es zum Iterieren verwendet wurde.

d) Wozu dient der Operator `&` in C. Auf was kann er angewendet werden.

e) Was berechnet in C der Operator `sizeof` für einen Parameter, mit dem ein Array übergeben wurde?