Programmiertechnik II
Klausur SS 2018

Angewandte Informatik Bachelor

Name	
Matrikelnummer	

Aufgabe	Punkte
1	9
2	12
3	12
4	15
Zwischen-summe	**48**

Aufgabe	Punkte
5	23
6	13
7	20
8	16
Summe	**120**
Note	

Aufgabe 1 (9 Punkte)

Die Klasse `Node` ist wie folgt definiert.

```
class Node {
     public Node next;
     public int[] data;

     public Node(Node p, int[] a) {
          this.next = p;
          this.data = a;
     }
}
```

a) Beschreiben Sie mit einem Speicherbelegungsbild für die Variable list, was durch folgende Anweisungen geleistet wird:

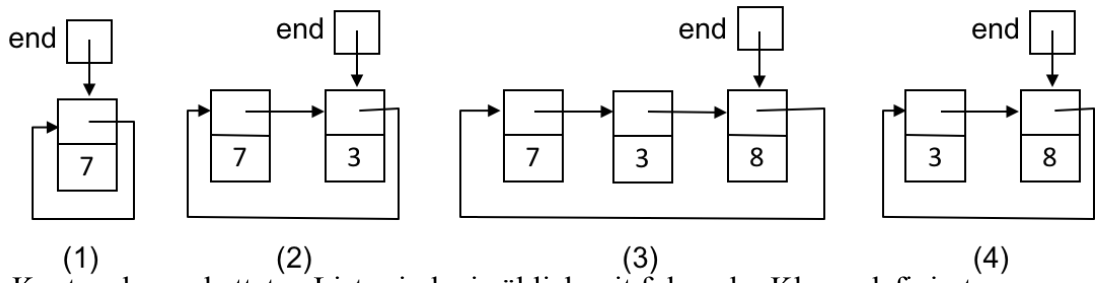
```
int[] a = {1,2};
Node list = new Node(null, a);
a = new int[]{4};
list = new Node(list, a);
list = new Node(list, new int[]{5,3,2});
```

b) Was wird auf die Konsole ausgegeben, wenn die Anweisungen aus a) und dann folgende Anweisungen durchgeführt werden.

```
for (Node p = list; p != null; p = p.next) {
     for (int x : p.data)
          System.out.print(x + ", ");
     System.out.println("");
}
```

Aufgabe 2 (12 Punkte)

Eine Schlange ist als verkettete Ringliste realisiert. Folgende Abbildung zeigt vier unterschiedliche Zustände einer Schlange. Z.B. stellt (3) eine Schlange mit den Elementen 7, 3 und 8 dar, wobei das vorderste Element 7 und das letzte Element 8 ist. `end` zeigt immer auf das letzte Element der Schlange.



(1) (2) (3) (4)

Die Knoten der verketteten Liste sind wie üblich mit folgender Klasse definiert.

```
class Node {
     public int data;
     public Node next;
     public Node(int d, Node n) {
          data = d;
          next = n;
     }
}
```

a) Schreiben Sie mit Benutzung von `end` eine (mehrere) Zuweisung(en), um die Schlange in den Zustand (1) zu bringen.

b) Schreiben Sie mit Benutzung von `end` eine (mehrere) Zuweisung(en), um die Schlange von Zustand (2) in (3) überzuführen.

c) Schreiben Sie mit Benutzung von `end` eine (mehrere) Zuweisung(en), um die Schlange von Zustand (3) in (4) überzuführen.

d) Schreiben Sie mit Benutzung von `end` eine Schleife zur Ausgabe einer Schlange. Die Schlange soll dabei unverändert bleiben. Bei Schlange (3) wäre die Ausgabe 7, 3, 8.

Aufgabe 3 QuickSort mit 3-Median-Strategie (12 Punkte)

Das 11-elementige Feld a = {5, 15, 2, 17, 2, 1, 3, 7, 4, 6, 10} wird mit Quicksort mit 3-Median-Strategie sortiert. Beschreiben Sie, wie sich dabei das Feld a ändert. Benutzen Sie eine tabellenartige Darstellung wie in der Vorlesung. Geben Sie außerdem die Aufrufstruktur von Quicksort an.

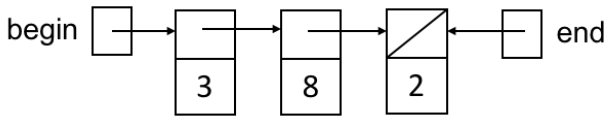
Die 3-Median-Strategie soll dabei wie folgt umgesetzt werden:
Sortieren Sie die 3 Zahlen a[li], a[m] und a[re] mit m = (li+re)/2 (a[li] ist das Element am linken Rand, a[m] ist das Element in der Mitte und a[re] ist das Element am rechten Rand). Vertauschen Sie dann a[m] mit a[re]. Die 3-Median-Strategie darf in einem Schritt durchgeführt werden (d.h. eine Zeile in der Tabelle).

Außerdem soll folgende Vereinfachung berücksichtigt werden:
Besteht das zu sortierende Teilfeld genau aus 2 Elementen, dann darf das Teilfeld durch einen einfachen Vertauschungsschritt sortiert werden.

5	15	2	17	2	1	3	7	4	6	10

Aufgabe 4 Linear verkettete Liste (15 Punkte)

Es soll eine Klasse für linear verkettete Listen (ohne Hilfskopfknoten) realisiert werden. Es wird eine Referenz auf den Anfang (begin) und eine Referenz auf das Ende (end) der Liste gespeichert.



Ergänzen Sie die Klasse auf der nächsten Seite um die folgenden Methoden.

a) contains(x) liefert true zurück, falls x in dieser Liste enthalten ist.

b) add(x) hängt ein Element x an das Ende dieser Liste ein. Implementieren Sie add ohne Schleife, indem Sie die Referenz end verwenden. Beachten Sie Spezialfälle.

c) add(l) fügt eine Kopie der Liste l an das Ende dieser Liste ein. Für add(l) wird eine Laufzeit von O(n) garantiert, wobei n die Anzahl der Elemente in der Liste l ist.

d) startsWith(l) prüft, ob die Liste l mit dem Anfang dieser Liste übereinstimmt. In folgendem Beispiel stimmt die Liste t mit dem Anfang der Liste s überein:

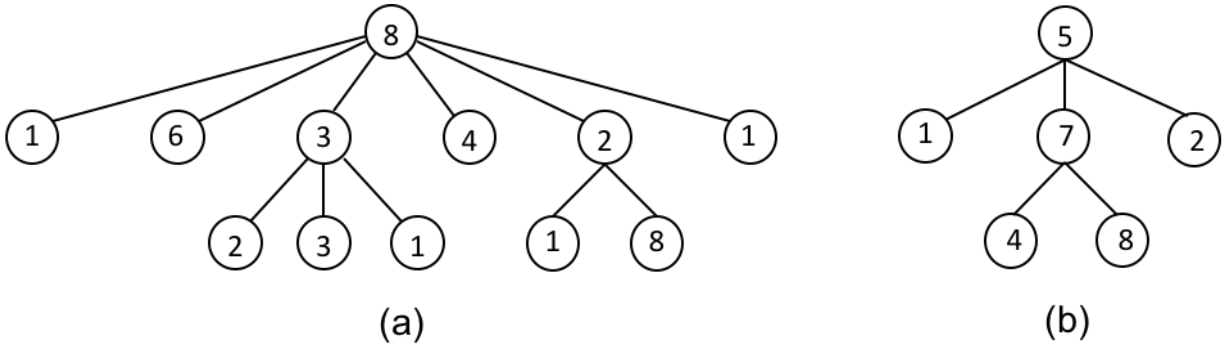
```
List s = new List();
s.add(1); s.add(2); s.add(3); ...   // s = 1,2,3,5,2,3,7

List t = new List();
t.add(1); t.add(2); t.add(3);       // t = 1,2,3

boolean b = s.startsWith(t);        // true
```

```
public class List {
      static private class Node {
            private int data;
            private Node next;
            private Node(Node p, int x) {this.data = x; this.next = p;}
      }
      private Node begin, end;

      public List() {
            begin = end = null;
      }


      public boolean contains(int x) {









      public void add(int x) {









      public void add(List l) {









      public boolean startsWith(List l) {









}
```

Aufgabe 5 Bäume (23 Punkte)

Die Klasse `Tree` definiert Bäume mit int-Daten in jedem Knoten und einer beliebigen Anzahl von Kindern. `root` ist die Wurzel des Baums.

In der Abbildung (a) und (b) sind zwei Beispielbäume gezeigt.



Ein Teil der Klasse `Tree` ist auf dieser und der nächsten Seite bereits vorgegeben. Lösen Sie folgende Aufgaben:

a) Die statische Methode `buildTree` erzeugt einen Baum. Stellen Sie den Baum grafisch dar.

b) Schreiben Sie eine Methode `prettyPrint`, die einen Baum auf der Konsole ausgibt, wobei jeder Knoten des Baums seiner Tiefe entsprechend eingerückt wird.

Beispielsweise ergibt sich beim Baum (b) die Ausgabe:

```
5
    1
    7
        4
        8
    2
```

c) Schreiben Sie eine Methode `collectEven`, die alle geraden Zahlen im Baum in einer PreOrder-Reihenfolge als Liste zurückliefert. `collectEven` ruft die private Methode `collectEvenR` auf, die den Baum rekursiv durchläuft und alle geraden Zahlen aufsammelt. Schreiben Sie beide Methoden.

d) Welche Liste gibt Ihre Methode `collectEven` für den in (a) gezeigten Baum zurück?

e) Schreiben Sie eine Methode `numberOfLeaves`, die die Anzahl der Blätter in einem Baum bestimmt. Beispielsweise hat der Baum (b) genau 4 Blätter.

```
class Tree {

   private static class Node {
      int data;
      List<Node> children;
      Node(int d) {
         data = d;
         children = new LinkedList<>();
      }
   }
   private Node root = null;
```

```
   public static Tree buildTree() {
      Tree t = new Tree();
      t.root = new Node(5);
      t.root.children.add(new Node(2));
      Node p = new Node(3);
      p.children.add(new Node(1));
      p.children.add(new Node(4));
      t.root.children.add(p);
      return t;
   }

   public void prettyPrint() {









   public int numberOfLeaves() {









   public List<Integer> collectEven() {









}
```

Aufgabe 6 Collection und Subtyping (13 Punkte)

a) Die Methode union vereinigt die beiden Mengen a und b und speichert das Ergebnis in c ab. Die Methode join berechnet den Schnitt der beiden Mengen a und b und speichert das Ergebnis in c ab. Definieren Sie beide Methoden.

```
public static <T> void union(Set<? extends T> a, Set<? extends T> b, Set<? super T> c) {




}

public static <T> void join(Set<? extends T> a, Set<? extends T> b, Set<? super T> c) {




}
```

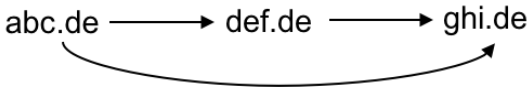
b) Definieren Sie in Java zwei einelementige Mengen a und b vom Typ Set. Und schreiben Sie einen Aufruf von union, der die beiden Mengen a und b zu einer Menge c vereinigt.

c) Welche Parametertypen sind beim Aufruf der Methode union syntaktisch erlaubt? Kennzeichnen Sie dazu in folgender Tabelle die korrekten Parametertypen mit „+“.

Parametertyp für a	Parametertyp für b	Parametertyp für c	Korrekt?
Set<Integer>	Set<Integer>	List<Integer>	
TreeSet<Double>	Set<Double>	SortedSet<Number>	
Set<String>	Set<Integer>>	Set<Object>	
Set<Object>	TreeSet<Integer>	Set<Integer>	

Aufgabe 7 Java Map (20 Punkte)

Web-Seiten mit ihren Links sollen in einer Link-Tabelle verwaltet werden. Beispielsweise hat „abc.de“ einen Link auf „def.de“ und „ghi.de“ und „def.de“ hat einen Link auf „ghi.de“.



In der Klasse `LinkTable` (s. nächste Seite) ist eine Instanzvariable `tab` vom Typ `Map` definiert. `tab` speichert für jede Web-Seite die Menge ihrer Links ab. Eine Web-Seite wird als String gespeichert.

Ergänzen Sie die Klasse auf der nächsten Seite um die folgenden Methoden.

a) Definieren Sie eine Methode `addLink(w1,w2)`, die einen Link von `w1` auf `w2` zur Link-Tabelle hinzufügt. Beispielsweise baut folgende Anweisungsfolge die in der Abbildung gezeigte Linkstruktur auf:

```
LinkTable tab = new LinkTable();
tab.addLink("abc.de", "def.de");
tab.addLink("abc.de", "ghi.de");
tab.addLink("def.de", "ghi.de");
```

b) Schreiben Sie eine Methode `getLinks(w)`, die die Menge aller verlinkten Web-Seiten zurückliefert. Z.B. liefert `getLinks("abc.de")` die Menge `{"def.de","ghi.de"}` und `getLinks("ghi.de")` die leere Menge zurück.

c) Schreiben Sie eine Methode `getSources(w)`, die die Menge alle Web-Seiten zurückliefert, die einen Link auf w haben. Z.B. liefert `getSources("ghi.de")` die Menge `{"abc.de","def.de"}` zurück.

d) Schreiben Sie eine Methode `print`, die alle Web-Seiten nach der Anzahl der ausgehenden Links sortiert und dann die Web-Seiten mit ihrer Link-Anzahl ausgibt. Beispielsweise gibt print für die in der Abbildung gezeigten Linkstruktur folgendes aus:

```
abc.de    2
def.de    1
ghi.de    0
```

```
public class LinkTable {
      private Map< String, Set<String> > tab = new TreeMap<>();
      void addLink(String w1, String w2) {









      }
      Set<String> getLinks(String w) {









      }
      Set<String> getSources(String w) {









      }
      void print() {









      }
}
```

Aufgabe 8 Java 8 (16 Punkte)

Gegeben ist die Klasse Pruefung und eine Liste prList von Prüfungen:

```
class Pruefung {
        public String name;     // Prüfungsteilnehmer
        public String fach;     // Prüfungsfach
        public double note;
        public Pruefung (String n, String f , double x) {name = n; fach = f; note = x;}
        public toString() {…}
}

List<Pruefung> prList = new LinkedList<>();
prList.add(new Pruefung("Mueller", "Prog2", 1.3));
prList.add(new Pruefung("Maier", "Prog2", 2.3));
prList.add(new Pruefung("Mueller", "Symo", 2.0));
// …
```

a) Definieren Sie ein Prädikat bestanden, das prüft, ob eine Prüfung bestanden wurde (Note ≤ 4.0).

b) Schreiben Sie eine foreach-Schleife, die mit Hilfe des Prädikats bestanden aus a) alle nicht bestandenen Prüfungen ausgibt.

c) Erzeugen Sie aus der Liste prList einen Strom und geben Sie mit Hilfe von Strom-Operationen die 3 besten Prüfungen in "Prog2" aus:

d) Erzeugen Sie aus der Liste prList einen Strom und berechnen Sie mit Hilfe von Strom-Operationen für den Prüfungsteilnehmer "Mueller"die Durchschnittsnote aller bestandenen Fächer.