Programmiertechnik II
Klausur WS 2019/20

Angewandte Informatik Bachelor

Name	
Matrikelnummer	

Aufgabe	Punkte
1	6
2	8
3	14
4	16
5	16
Zwischen-summe	**60**

Aufgabe	Punkte
6	30
7	12
8	18
Summe	**120**
Note	

Aufgabe 1 *(6 Punkte)*

Beschreiben Sie mit einem Speicherbelegungsbild (Referenzen als Pfeile!), was durch die main-Methode geleistet wird. Es genügt das Speicherbelegungsbild anzugeben, nachdem alle Anweisungen der main()-Methode ausgeführt worden sind.

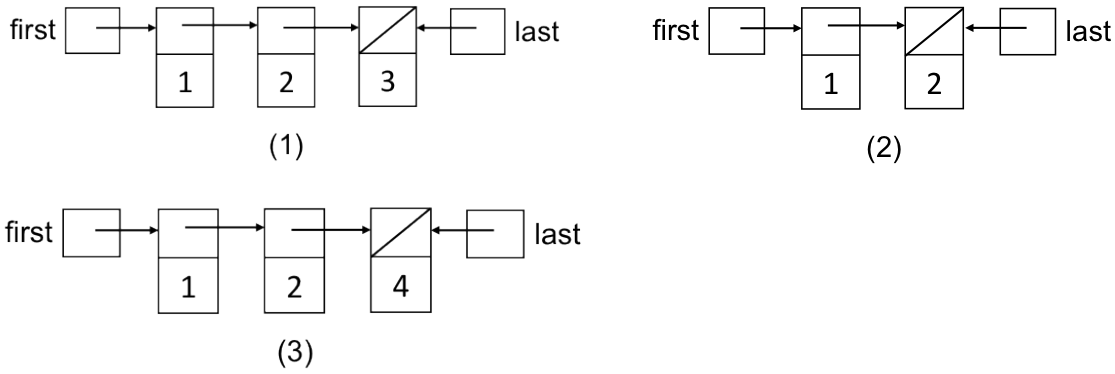
```
static class Node {
   Node next;
   int data;

   Node(int x, Node p) {
      next = p;
      data = x;
}

public static void main(String[] a) {
   Node p = new Node(5, null);
   p.next = new Node(7, p.next);
   p = new Node(8, p);
   Node q = p.next;
   q.next = new Node(3, q.next);
   q = new Node(9, null);
}
```

Aufgabe 2 ***(8 Punkte)***

Für Knoten einer linear verketteten Liste sei folgende Klasse definiert:

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

Die folgende Abbildung zeigt drei verschiedene Zustände einer linear verketteten List. **`first`** zeigt auf den ersten Knoten und **`last`** zeigt auf den letzten Knoten.



a) Schreiben Sie eine Folge von Anweisungen, die die Liste von Zustand (1) in Zustand (2) überführt. Verwenden Sie keine Schleife und benutzen Sie nur die Variablen **`first`** und **`last`**.

b) Schreiben Sie eine Folge von Anweisungen, die die Liste von Zustand (2) in Zustand (3) überführt. Verwenden Sie keine Schleife und benutzen Sie nur die Variable **`last`**.

Aufgabe 3 QuickSort mit 3-Median-Strategie (14 Punkte)

Das 12-elementige Feld a = {16, 12, 3, 20, 10, 13, 8, 1, 4, 11, 6, 8} 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 nur aus 2 oder 3 Elementen, dann darf das Teilfeld durch einfache Vertauschungsschritte sortiert werden. Die Vertauschungen dürfen in einem Schritt durchgeführt werden (d.h. eine Zeile in der Tabelle).

16	12	3	20	10	13	8	1	4	11	6	8

Aufgabe 4 Linear verkettete Liste (16 Punkte)

Eine Liste von Variablennamen mit ihren int-Werten soll in einer linear verketteten Liste ohne Hilfskopfknoten verwaltet werden. Wird auf einen Variablenamen zugegriffen, so rückt der entsprechende Eintrag an den Listenanfang. Damit kann ein späterer Zugriff auf den gleichen Variabennamen effizienter erfolgen (most recently used name list). Definieren Sie folgende Methoden:

a) `get(n)` liefert den Wert der Variablen mit Name `n` zurück. Falls der Name `n` nicht vorkommt wird `null` zurückgeliefert. Falls `n` vorkommt, dann rückt der entsprechende Eintrag an den Anfang der Liste. (9 Punkte)

b) `add(n,v)` fügt einen neuen Eintrag mit Name `n` und Wert `v` an den Anfang der Liste, falls n in der Liste nicht vorkommt. Ansonsten wird der alte Wert von `n` mit `v` überschrieben und der entsprechende Eintrag rückt an den Anfang der List. (5 Punkte)

c) Welche Komplexität hat die Laufzeit von get und von add (O-Notation). Gehen Sie davon aus, dass Ihre Liste n Einträge enthält. (2 Punkte)

```
public class NameList {
   static private class Node {
        private String name;   // Variabenname
        private int value;     // Wert der Variablen
        private Node next;
        private Node(String n, int v, Node p) {
             name=n; value=v; next=p;
        }
   }
   private Node head = null;









}
```

Aufgabe 5 Binärer Suchbaum (16 Punkte)

Die folgende (unvollständige) Klassendefinition definiert einen binären Suchbaum.

a) Schreiben Sie eine rekursive Methode (ohne Schleifen) `equalsR`, die prüft, ob zwei binäre Suchbäume identisch sind (gleiche Struktur und gleiche Zahlen) (7 Punkte).

b) Schreiben Sie eine rekursive Methode (ohne Schleifen) `collectR`, die alle Zahlen im Suchbaum, die das Prädikat `pred` erfüllen, in eine Liste aufsammelt (6 Punkte).

c) Ist die von `collect` zurückgelieferte Liste sortiert oder nicht sortiert? Begründen Sie (3 Punkte).

```
public class BinarySearchTree {

   static class Node {
        int data;
        Node left;
        Node right;
   }
   private Node root = null;
   // ...

   public boolean equals(BinarySearchTree bst) {
        if (bst == null)
             return root == null;
        else
             return equalsR(root, bst.root);
   }
```

```
   public List<Integer> collect(Predicate<Integer> pred) {
        List<Integer> l = new LinkedList<>();
        collectR(root, l, pred);
        return l;
   }
```

```
}
```

Aufgabe 6 Collection (30 Punkte)

In einem Objekt der Klasse Pruefung wird festgehalten, dass ein Student ein Fach mit einer bestimmten Prüfungsnummer absolviert hat:

```
class Pruefung {
        public String name;      // Name des Studenten
        public int pruefNr;      // Prüfungsnummer des Fachs
        public Pruefung (String n, int pnr) {name = n; pruefNr = pnr;}
        public String toString() { return "name=" + name + ", pruefNr =" + pruefNr; }
}
```

Schreiben Sie die Lösung der folgenden Teilaufgaben auf die nächste Seite!

a) Definieren Sie eine Methode list2Map(pruefList), der eine Liste pruefList von bestandenen Prüfungen übergeben wird und die eine Map zurückliefert, die jedem Student die Menge der Prüfungsnummern aller bestandenen Fächer zuordnet. (6 Punkte)

b) Definieren Sie eine Methode map2List(s2n), der eine Map s2n übergeben wird, die jedem Student die Menge der Prüfungsnummern aller bestandenen Fächer zuordnet, und die eine Liste der bestandenen Prüfungen zurückliefert. map2List ist die inverse Funktion zu list2Map. (7 Punkte)

c) Definieren Sie eine Methode invertMap(s2n), der eine Map s2n übergeben wird, die jedem Student die Menge der Prüfungsnummern aller bestandenen Fächer zuordnet, und die eine Map zurückliefert, die jeder Prüfungsnummer die Menge aller Studenten zuordnet, die dieses Fach bestanden haben. invertMap(s2n) liefert die invertierte Map zurück. (8 Punkte)

d) Was gibt die main-Funktion auf die Konsole aus? (5 Punkte)

```
public static void main(String[ ] args) {
        List<Pruefung> pruefList = new ArrayList<>();
        pruefList.add(new Pruefung("Maier",14100));
        pruefList.add(new Pruefung("Mueller",14150));
        pruefList.add(new Pruefung("Anton",14120));
        pruefList.add(new Pruefung("Zimmer",14100));
        pruefList.add(new Pruefung("Maier",14150));
        pruefList.add(new Pruefung("Mueller",14160));
        pruefList.add(new Pruefung("Mueller",14120));
        pruefList.add(new Pruefung("Mueller",14100));
        pruefList.add(new Pruefung("Anton",14100));

        Map<String, Set<Integer>> s2n = list2Map(pruefList);
        System.out.println(s2n);

        List<Pruefung> l = map2List(s2n);
        for (int i = 0; i < 5; i++)
                System.out.println(l.get(i));

        Map<Integer, Set<String>> n2s = invertMap(s2n);
        System.out.println(n2s);
}
```

e) Welche Komplexität hat die Laufzeit von list2Map(pruefList) (O-Notation) aus Aufgabe a). Gehen Sie davon aus, dass in pruefList n unterschiedliche Studenten vorkommen. Die Anzahl verschiedener Prüfungen kann als konstant angesehen werden. Gleiche Objekte kommen in pruefList nicht mehrfach vor. Begründen Sie kurz Ihre Antwort kurz. (4 Punkte)

```
public static Map<String, Set<Integer>> list2Map(List<Pruefung> l){




}
public static List<Pruefung> map2List(Map<String, Set<Integer>> s2n){




}
public static Map<Integer, Set<String>> invertMap(Map<String, Set<Integer>> s2n){




}
```

Ausgabe von main():

Komplexität von list2Map(pruefList):

Aufgabe 7 Subtyping (12 Punkte)

Es werden folgende Variablen definiert, wobei die Initialisierungen weggelassen sind.

```
List<String> listStr;
List<Object> listObj;
Set<Integer> setInt;
Queue<Double> queueDb;
Collection<Double> colDb;
Collection<Number> colNb;
Collection<Integer> colInt;
Collection<Object> colObj;
```

Die statische Methode `copy` aus der Klasse Collections kopiert die Liste `src` in die Liste `dest`.

```
static <T> void copy(List<? super T> dest, List<? extends T> src)
```

Geben Sie in folgender Tabelle für jeden Aufruf an, ob er korrekt ist („+“) oder ob er nicht korrekt ist („-“). Falsche Antworten geben Abzüge!

`listStr.containsAll(colNb);`	
`colInt.addAll(colNb);`	
`listStr.removeAll(colObj);`	
`colNb.addAll(setInt);`	
`listObj.add(queueDb);`	
`setInt.addAll(colInt);`	
`colDb.containsAll(setInt);`	
`listObj.addAll(colDb);`	
`colDb.addAll(setInt);`	
`Collections.copy(listObj,listStr);`	
`Collections.copy(listStr,listObj);`	
`Collections.copy(colObj,listObj);`	

Aufgabe 8 Lambda-Ausdrücke und Ströme (18 Punkte)

Gegeben ist eine Klasse Pruefung und eine Liste von Prüfungen pruefListe:

```
class Pruefung {
        public String name;        // Name des Student
        public String fach;        // Fach
        public double note;        // Note
        public Pruefung (String s, String f , double n) { name = n; fach = f; note = n;}
}
List< Pruefung > pruefListe = new LinkedList<>();
pruefListe.add(new Pruefung ("Maier","Prog2",1.3));
pruefListe.add(new Pruefung ("Mueller","Prog2",5.0));
pruefListe.add(new Pruefung ("Mueller","Rarc",2.0));
// ...
```

a) Definieren Sie ein Prädikat (mit Typ), das prüft ob eine Prüfung bestanden ist (Note ≤ 4.0). (2 Punkte)

b) Definieren Sie einen Comparator (mit Typ) für Prüfungen als Lambda-Ausdruck, indem die Noten numerisch verglichen werden. (3 Punkte)

c) Erzeugen Sie aus pruefListe einen Strom und berechnen Sie mit Hilfe von Strom-Operationen die Durchschnittnote für alle bestandenen Prüfungen im Fach „Prog2“ aus. Benutzen Sie das Prädikat bestanden aus a). (5 Punkte)

d) Erzeugen Sie aus pruefListe einen Strom und geben Sie die von „Maier“ absolvierten Prüfungen aus. Die Ausgabe ist alphabetisch nach dem Fach sortiert und besteht nur aus Fach und Note. (4 Punkte)

e) Erzeugen Sie aus pruefListe einen Strom und sammeln Sie die Namen aller Studenten in einer sortierten Liste. Hinweis: collect-Aufruf verwenden. (4 Punkte)