Lehrstuhl für Datenbanksysteme
Fakultät für Informatik
Technische Universität München

Esolution
Sticker mit SRID hier einkleben

Hinweise zur Personalisierung:

- Ihre Prüfung wird bei der Anwesenheitskontrolle durch Aufkleben eines Codes personalisiert.
- Dieser enthält lediglich eine fortlaufende Nummer, welche auch auf der Anwesenheitsliste neben dem Unterschriftenfeld vermerkt ist.
- Diese wird als Pseudonym verwendet, um eine eindeutige Zuordnung Ihrer Prüfung zu ermöglichen.

Probeklausur Einführung in die Informatik 2 für Ingenieure

Klausur: IN8012 / Probeklausur **Datum:** Sonntag, 19. Juli 2020
Prüfer: Prof. Dr. Alfons Kemper **Uhrzeit:** 20:30 – 23:30

	A 1	A 2	A 3	A 4	A 5	A 6	A 7
I							
II							

Bearbeitungshinweise

- Diese Klausur umfasst **11 Seiten** mit insgesamt **7 Aufgaben**.
  Bitte kontrollieren Sie jetzt, dass Sie eine vollständige Angabe erhalten haben.
- Die Gesamtpunktzahl in dieser Prüfung beträgt 60 Punkte.
- Das Heraustrennen von Seiten aus der Prüfung ist untersagt.
- Als Hilfsmittel sind zugelassen:
  - **Open Book**: Alle Materialien aus der Vorlesung oder anderen Quellen sind als Hilsmittel zugelassen.
  - Zur Bearbeitung der Klausur darf **keine** Hilfe von anderen Personen verwendet werden.
- Sie können dieses Dokument ausdrucken, bearbeiten, digitalisieren und hochladen **oder** das Dokument digital ausfüllen. Verwenden Sie in diesem Fall **nicht** die Annotations- oder Kommentarfunktion Ihres PDF-Programms, sondern die dazu vorgesehenen Texteingabefelder.
- Alle Lösungen **müssen** in die dazu vorgesehenen Felder eingetragen werden. Insbesondere können von Ihnen hinzugefügte Blätter **nicht** gewertet werden.
- Schreiben Sie weder in roter noch grüner Farbe.
- Beschriften Sie **nicht** die Bewertungsfelder.
- Verwenden Sie, falls nicht anders angegeben, die Algorithmen, die in der Vorlesung und Übung verwendet wurden. Selbstentwickelte Algorithmen werden im Allgemeinen nicht gewertet.
- Falls nicht anders angegeben, so lösen Sie Aufgaben zum Thema SQL nur mit den Mitteln des SQL-92 Standards und/oder den in der Vorlesung vorgestellten Konstrukten. Insbesondere dürfen keine nicht standardisierten Konstrukte verwendet werden.
- Falls Sie Zwischenergebnisse oder Lösungswege angeben, machen Sie das finale Ergebnis stets als solches kenntlich.
- Achten Sie beim Abgeben darauf, dass die QR-Codes klar lesbar sind.

Hörsaal verlassen von _____ bis _____ / Vorzeitige Abgabe um _____

Aufgabe 1 Java (8 Punkte)

Betrachten Sie die folgende Java-Methode:

```
public static int meine_methode(List<Integer> input) {
  short a = 0;
  int c = 0;
  for (int i = 0; i < input.size(); i++) {
    a += input.get(i);
    c++;
  }
  return a / c;
}
```

0 1 2 3 4 5

a) Was berechnet diese Methode? Beschreiben Sie zwei Fälle, in denen die Methode sich unerwartet verhält!

Berechnet den Durchschnitt der Werte in der Eingabeliste. Unerwartet:

- Die variable `a` ist ein `short`, dies kann leicht zu einem Überlauf führen.
- Der Parameter "`input`" könnte `null` sein, was zu einer Ausnahme führen würde.
- Ein Element der Eingabeliste könnte `null` sein, was zu einer Ausnahme führen würde.
- Die Methode gibt ein `int` zurück. Also sind die Durchschnittswerte immer abgerundet.

0 1 2 3

b) Vervollständigen Sie das folgende Programmfragment so, dass als Ergebnis von `meine_methode(zahlen)` die Zahl 3 ausgeben wird.

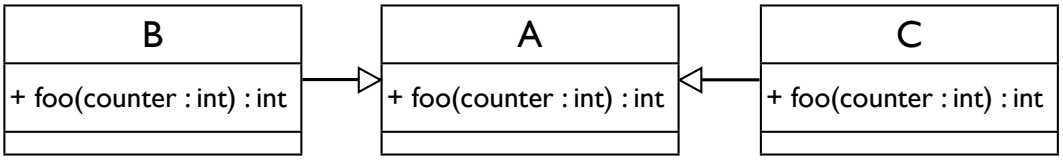
```
List<Integer> zahlen = new ArrayList<Integer>();
zahlen.add(100);
```

```
// Hier gibt es natürlich viele Möglichkeiten
zahlen.add(-94);
```

```
System.out.print("Das Ergebnis ist: ");
System.out.println(meine_methode(zahlen));
```

Aufgabe 2 Java: Generalisierung und Spezialisierung (4 Punkte)

Das folgende UML Diagramm beschreibt eine einfache Klassen Hierarchie in Java:

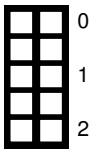


Betrachten die folgenden Java-Programmfragmente:

- Falls diese **einen Fehler** enthalten, beschreiben Sie den Fehler.
- Falls ein Fragment **keinen Fehler** enthält, geben Sie an aus welcher der Klassen die `foo` Methode ausgeführt wird und ob dynamisches binden benötigt wird.

a)

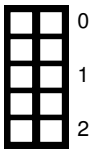
```
A a = new B();
a.foo();
```



Kein Fehler, durch dynamisches binden wird B::foo() ausgeführt.

b)

```
A a = new B();
C c = (C) a;
c.foo();
```



Laufzeitfehler in Zeile 2. Ein Objekt vom Typen `B` kann icht auf Type `C` ge-castet werden.

Aufgabe 3 Java (12 Punkte)

0 1 2 3 4 5 6 7 8 9 10 11 12

Implementieren Sie die Lookup-Methode einer Hash-Table in Java, die zur Kollisionsbehandlung **lineares Probing** verwendet. Die Hash-Table soll effizientes Suchen nach `String`-Werten unterstützen. Den `String`-Werten ist jeweils ein `int`-Wert zugeordnet ist. Ergänzen Sie dazu die Methode `lookup()` in der folgenden Klasse.
Hinweise:

- Jedes Objekt in Java hat eine Methode `hashCode()`, die einen `int`-Wert zurückgibt
- Gehen Sie davon aus, dass alle Einträge in der Tabelle (`table`) ein korrekt konstruiertes `Entry`-Objekt enthalten
- Ob ein Eintrag der Hash-Table leer ist, kann mit dem Attribut `is_empty` abgefragt werden
- Wenn ein Wert nicht gefunden wird, soll `lookup()` den Wert -1 zurückgeben
- `lookup()` soll auch dann funktionieren, wenn die Hash-Table voll ist, d.h. alle Einträge nicht leer sind

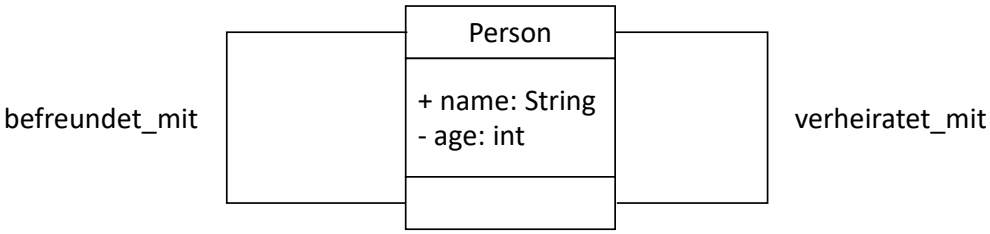
```
class HashTableWithLinearProbing {
    class Entry {
        bool is_empty;
        String key;
        int value;
    }

    Entry[] table;

    public int lookup(String key) {
```

```
int hash_code = key.hashCode();
int position = hash_code % table.length;
int initial_position = position;

while (!table[position].is_empty) {
  // Check if this is our key
  if (table[position].key.compareTo(key) == 0) {
    return table[position].value;
  }

  // Iterate
  position = (position + 1) % table.length;

  // Did one full scan of the table
  if (position == initial_position) {
    break;
  }
}

return -1;
```

```
    }
}
```

Aufgabe 4 UML (4 Punkte)

Betrachten Sie das Folgende UML Diagramm:



Übersetzten Sie das Model möglichst genau nach Java! Erstellen Sie die entsprechende `Person` Klasse. Diese Klasse soll alle in UML beschriebenen Klassenvariablen und Beziehnungen speichern können.

0
1
2
3
4

```java
class Person {
  public String name;
  private int age;
  private List<Person> befreundet_mit;
  private Person verheiratet_mit;
}
```

Aufgabe 5 SQL (20 Punkte)

Gegeben das bekannte Universitätsschema:

Studenten	:	{[MatrNr : integer, Name : string, Semester : integer]}
Vorlesungen	:	{[VorlNr : integer, Titel : string, SWS : integer, gelesenVon : integer]}
Professoren	:	{[PersNr : integer, Name : string, Rang : string, Raum : integer]}
Assistenten	:	{[PersNr : integer, Name : string, Fachgebiet : string, Boss : integer]}
hören	:	{[MatrNr : integer, VorlNr : integer]}
voraussetzen	:	{[Vorgänger : integer, Nachfolger : integer]}
prüfen	:	{[MatrNr : integer, VorlNr : integer, PersNr : integer, Note : integer]}

Beantworten Sie in **SQL**:

0 1 2

a) Welche Vorlesungen hält der Professor „Sokrates“? (Nur Titel der Vorlesungen ausgeben)

```
SELECT v.Titel
FROM Professoren p, Vorlesungen v
WHERE p.PersNr = v.gelesenVon
AND p.Name = 'Sokrates'
```

0 1 2 3 4

b) Geben Sie die Namen aller Studenten aus, die keine Vorlesung mit mehr als 4 SWS hören. (Spalten in der Ausgabe: Name der Studenten)

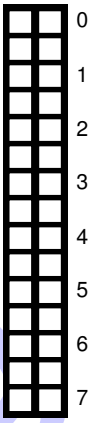
```
SELECT s.name
FROM Studenten s
WHERE not exists (SELECT *
   FROM hoeren h, Vorlesungen v
   WHERE s.matrNr = h.matrnNr
   and h.matrNr = v.vorlNr
   and v.sws > 4
)
```

0 1 2 3 4 5 6 7

c) Geben Sie für jede Vorlesung an, wie viele direkte voraussetzende und wie viele direkte nachfolgende Vorlesungen sie hat. Beachten Sie dabei, dass Vorlesungen ohne Voraussetzungen oder Nachfolger auch ausgegeben werden müssen. (Spalten in der Ausgabe: VorlNr, Titel, Anzahl Voraussetzungen, Anzahl Nachfolger)

```
SELECT v.vorlNr, v.Titel, count(distinct vor.Vorgaenger),
       count(distinct nach.Nachfolger)
FROM
Vorlesungen v LEFT OUTER JOIN
voraussetzen vor ON (v.VorlNr = vor.Nachfolger) LEFT OUTER JOIN
voraussetzen nach ON (v.VorlNr = nach.Vorgaenger)
GROUP BY v.vorlNr, v.Titel
```

d) Finden Sie die Studenten, die in allen Vorlesungen, die Sie bei „Kant“ gehört haben, auch eine Prüfung geschrieben haben. (Nur die Namen der Studenten ausgeben)




```
SELECT s.Name
FROM Studenten s
WHERE NOT EXISTS (
    SELECT *
    FROM hoeren h, Vorlesungen v, Professoren p
    WHERE h.matrnr = s.matrnr
    AND h.vorlnr = v.vorlnr
    AND v.gelesenVon = p.PersNr
    AND p.Name = 'Kant'
    AND NOT EXISTS (
        SELECT *
        FROM pruefen p
        WHERE p.MatrNr = s.MatrNr
        AND p.VorlNr = v.VorlNr
    )
)
```

Aufgabe 6 Datenstrukturen (8 Punkte)

0 1 2

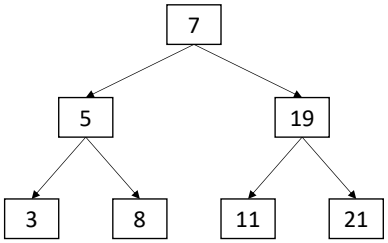
a) Nennen Sie einen Vorteil und einen Nachteil von AVL-Bäumen gegenüber Hash-Tables.

- Vorteil: AVL-Bäume erlauben Bereichsanfragen
- Nachteil: Suche in Hash-Table ist $\mathcal{O}(1)$, in AVL-Baum aber $\mathcal{O}(\log(n))$

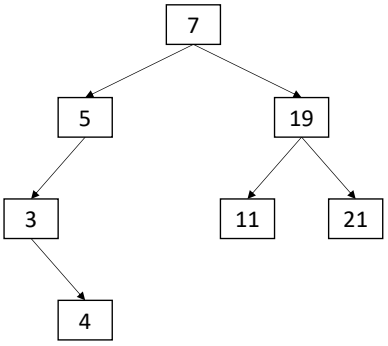
0 1 2 3 4 5 6

b) Betrachten Sie die folgenden drei AVL-Bäume (**A**, **B** und **C**).

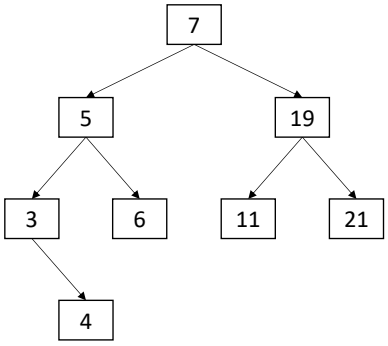
Baum **A**



Baum **B**



Baum **C**

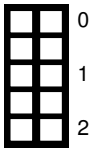


Zwei der drei Bäume verletzten die in der Vorlesung besprochenen AVL-Baum Eigenschaften. Identifizieren Sie diese und beschreiben Sie kurz was falsch ist und an welchem Knoten der Fehler auftritt.

A und B sind falsch.
Baum A ist nicht sortiert. Knoten “8” darf nicht links von der Wurzel “7” sein.
Baum B ist in Knoten “5” nicht balanziert (2).

Aufgabe 7 Datenbankenwissen (4 Punkte)

a) Nennen und erklären Sie kurz einen der Unterschiede zwischen Relationaler Algebra und SQL.



SQL ist eine deklarative Anfragesprache die von der Datenbank intern durch relationale Algebra abgearbeitet wird. Damit ist SQL die Sprache mit der Benutzter mit der Datenbank kommunizieren und relationale Algebra die mathematische Grundlage für die Anfragebearbeitung und Anfrageoptimierung.

b) Wie kann ein Anti-Join in SQL ausgedrückt werden.

Durch ein `not exists`.

[Optional] Was ist Ihrer Meinung nach das schlimmste Feature in Java (Begründung).

Man bekommt nicht einfach den Speicherverbrauch von einem Objekt und String-Pools.

Zusätzlicher Platz für Lösungen. Markieren Sie deutlich die Zuordnung zur jeweiligen Teilaufgabe. Vergessen Sie nicht, ungültige Lösungen zu streichen.