Name:	**Studiengang:** ☐ **B.A.** | ☐ **M.A.**
Vorname:	**Matrikelnummer:**
Studienfächer:	**Fachsemester:**

Allgemeine Hinweise:

1. Überprüfen Sie bitte, ob Sie alle Seiten der Klausurangabe vollständig erhalten haben (Gesamtzahl: **8**)

2. **Bearbeitungszeit: 90 Minuten**, maximal erreichbare **Punktzahl: 72**. Die jeweils erreichbare Punktzahl ist bei jeder Frage angegeben. Bitte teilen Sie Ihre Arbeitszeit entsprechend ein.

3. Denken Sie daran, die Daten oben einzutragen, **bevor** Sie mit der Bearbeitung beginnen.

4. Verwenden Sie für die Beantwortung aller Fragen diese Klausurangabe. Sie können bei Bedarf zusätzliche Blätter anfordern. Wenden Sie sich dafür bitte an die Aufsichtsführenden. Bitte geben Sie in jedem Fall an, auf welche Frage sich die Lösung jeweils bezieht. Bei Multiple-Choice-Fragen treffen Sie bitte die Auswahl Ihrer Antworten ebenfalls auf der Klausurangabe.

5. Benutzen Sie keine Bleistifte, keine rot schreibenden Stifte und kein TippEx, o.ä.

6. Zugelassene Hilfsmittel: **KEINE**

7. Mobiltelefone sowie Computer am Arbeitsplatz - auch ausgeschaltet - sind **nicht zugelassen**.

8. Bitte legen Sie Lichtbildausweis und Studierendenausweis gut sichtbar vor sich, damit Ihre Identität möglichst störungsfrei überprüft werden kann.

9. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung ergeben also 0 Punkte.

10. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Aufsichtsführenden. Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für alle Teilnehmer durchgegeben.

TEIL I) XML

I.1 Wahr oder falsch? (10 Punkte)

Beurteilen Sie für jede der folgenden Aussagen, ob diese Wahr oder Falsch ist. Bitte beachten Sie: Falsch angekreuzte Antworten führen zu Punktabzug, wobei eine Punktzahl kleiner als Null nicht erreicht werden kann.

Wahr		Falsch
☐	Wohlgeformte XML-Dokumente sind immer auch valide.	☐
☐	XML-Tags dürfen Leerzeichen enthalten.	☐
☐	Elementbezeichner dürfen das Wort xml enthalten.	☐
☐	XML-Parser erfordern die Wohlgeformtheit und Validität eines XML-Dokuments	☐
☐	Eine leere Datei ist ein wohlgeformtes XML-Dokument.	☐

I.2 Markieren Sie im nachstehenden XML-Dokument 4 Fehler, welche das Dokument nicht wohlgeformt sein lassen und korrigieren Sie diese. (8 Punkte)

```
1  <person id=1>
2  <vorname>Alice</vorname>
3  <nachname>Wunderland</nachname>
4  <geburtsdatum><geburtsdatum>
5  </person>
6  <person id="2">
7  <vorname>Bob</vorname>
8  <nachname>Bobson</nachname>
9  <geburtsdatum/>
10 <person id='3'>
11 </person>
12 <vorname>Clerk</vorname>
13 <nachname>Clerksen</nachname>
14 <geburtsdatum></geburtsdatum>
15 </person>
```

I.3 XPath (4 Punkte)

Gegeben sei folgendes XML-Dokument:

```
<planes>
    <plane>
        <year>1980</year>
        <model>Samplemodel</model>
        <color>yellow</color>
    </plane>
    <plane>
        <year>1990</year>
        <model>Airbus</model>
        <color>red</color>
    </plane>
    <plane>
        <year>2000</year>
        <model>Boeing</model>
        <color>blue</color>
    </plane>
</planes>
```

Schreiben Sie einen XPath-Ausdruck, der alle plane-Elemente selektiert.

Schreiben Sie einen XPath-Ausdruck, der das erste plane-Element selektiert.

I.4 XML vs. JSON (6 Punkte)

Nennen Sie drei wesentliche Unterschiede zwischen XML und JSON.

I.5 XML in JSON umwandeln (4 Punkte)

Wandeln Sie nachstehendes XML-Dokument in das JSON-Format um:

```
<catalog>
    <cd>
        <title>A Night at the Opera</title>
        <artist>Queen</artist>
        <country>UK</country>
        <year>1975</year>
    </cd>
    <cd>
        <title>Jazz ist anders</title>
        <artist>Die Ärzte</artist>
        <country>Germany</country>
        <year>2007</year>
    </cd>
</catalog>
```

I.6 XQuery (4 Punkte)

Gegeben sei nachstehender XQuery-Ausdruck:

```
for $i in (1,2), $j in (3,4)
return <zahl><i>{$i}</i><j>{$j}</j></zahl>
```

Welche Ausgabe liefert obiger XQuery-Ausdruck?

I.7 XML und XML-Schema (8 Punkte)

Ergänzen Sie die nachstehende XML- und XML-Schema-Datei jeweils so, dass sie wohlgeformt und valide sind. Stellen, an denen etwas ergänzt werden muss, sind durch mind. einen Unterstrich gekennzeichnet.

```
<?xml version="1.0" encoding="_____"?>
<gerichte>
    <gericht id="1">
        <name>Salat</name>
        <preis>2.50</preis>
        <fett>19.9<_____fett>
    </gericht>
    <gericht _____>
        <name>Currywurst</name>
        <preis>3.50</preis>
        <zucker>2.5</zucker>
    </gericht>
</gerichte>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://meinegerichte.de">
<xs:element name="gerichte">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="_____" maxOccurs="unbounded">
                <xs:complexType>
                        <xs:sequence>
                            <xs:element name="name" type="xs:string"/>
                            <xs:element name="preis" type="xs:string"/>
                            _____
                                <xs:element name="fett" type="xs:decimal">

                                _____
                                <xs:element name="zucker" type="xs:decimal"/>
                            _____
                        </xs:sequence>
                    <xs:attribute name="id" type=_____/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

I.8 XML-Schema (10 Punkte)

Erstellen Sie zu folgender XML-Datei eine XML-Schema-Datei, die es ermöglicht, die gegebene XML-Datei zu validieren. Verwenden Sie geeignete Datentypen!

```
<students>
    <student id="1">
        <name>
            <firstname>Alison</firstname>
            <lastname>Abraham</lastname>
        </name>
        <age>19</age>
    </student>
    <student id="2">
        <name>
            <firstname>Brian</firstname>
            <lastname>Baker</lastname>
        </name>
        <age>22</age>
    </student>
    <student id="3">
        <name>
            <firstname>Frank</firstname>
            <lastname>Fraser</lastname>
        </name>
        <age>30</age>
    </student>
</students>
```

I.9 XQuery (6 Punkte)

Gegeben sei nachstehendes XML-Dokument „store.xml“:

```xml
<?xml version="1.0" encoding="utf-8"?>
<store>
  <customers>
    <customer id="1">
      <firstname>Sam</firstname>
      <lastname>Graham</lastname>
    </customer>
    <customer id="2">
      <firstname>Andy</firstname>
      <lastname>Mack</lastname>
    </customer>
  </customers>
  <orders>
    <order id="1">
      <customerid>1</customerid>
      <price>200.00</price>
    </order>
    <order id="2">
      <customerid>1</customerid>
      <price>50.00</price>
    </order>
    <order id="3">
      <customerid>2</customerid>
      <price>100.00</price>
    </order>
  </orders>
</store>
```

Aus dieser XML-Datei möchten Sie mit Hilfe einer XQuery das folgende Ergebnis erzielen:

```xml
<customer_orders>
  <customer_order>
    <firstname>Sam</firstname>
    <lastname>Graham</lastname>
    <order id="1">
      <price>200.00</price>
    </order>
    <order id="2">
      <price>50.00</price>
    </order>
  </customer_order>
  <customer_order>
    <firstname>Andy</firstname>
    <lastname>Mack</lastname>
    <order id="3">
      <price>100.00</price>
    </order>
  </customer_order>
</customer_orders>
```

Welche der nachstehenden XQueries ist hierfür geeignet? (Bitte markieren Sie Ihre Antwort mit einem **X** im Kasten)

```
<customer_orders>
  {
    for $customer in doc("store.xml")//customer
    return
    <customer_order>
      {
        $customer/firstname,
        $customer/lastname,
        for $order in doc("store.xml")//order[$customer/@id = customerid]
        return
        <order id="{$order/@id}">
        {$order/price}
        </order>
      }
    </customer_order>
  }
</customer_orders>
```

```
<customer_orders>
  {
    for $customer in doc("store.xml")//customer
    return
    <customer_order>
      {
        $customer/firstname,
        $customer/lastname,
        for $order in doc("store.xml")//order[$customer/@id = customerid]
        return
        <order id={$order/@id}>
        {$order/price}
        </order>
      }
    </customer_order>
  }
</customer_orders>
```

```
<customer_orders>
  {
    for $customer in doc("store.xml")//customer
    return
    <customer_order>

        $customer/firstname,
        $customer/lastname,
        for $order in doc("store.xml")//order[$customer/@id = customerid]
        return
        <order id="{$order/@id}">
        {$order/price}
        </order>

    </customer_order>
  }
</customer_orders>
```

```
<customer_orders>
  {
    for $customer in doc("store.xml")//customer
    return
    <customer_order>
      {
        $customer/firstname,
        $customer/lastname,
        for $order in doc("store.xml")//order[$customer[@id] = customerid]
        return
        <order id="{$order/@id}">
        {$order/price}
        </order>
      }
    </customer_order>
  }
</customer_orders>
```

Teil II) JAVA, JSON und XML

II.1 (4 Punkte)

In welcher/n der folgenden Situationen ist es sinnvoll, einen SAX-Parser zu benutzen? Bitte beachten Sie: Falsche Antworten führen zu Punktabzug.

☐ Falls das XML-Dokument sehr groß ist.

☐ Falls das XML-Dokument lediglich sequentiell verarbeitet werden soll.

☐ Falls auf bestimmte Teile des XML-Dokuments wiederholt zugegriffen werden muss.

II.2 (4 Punkte)

Welche der folgenden Aussage(n) über SAX-Parser ist/sind korrekt? Bitte beachten Sie: Falsche Antworten führen zu Punktabzug.

☐ SAX ist ein Streaming-Interface für XML.

☐ SAX-Parser feuern Events für Start- und Endtags.

☐ Keine der obigen Aussagen ist korrekt.

II.3 (4 Punkte)

Welche der folgenden Aussage(n) über JDOM-Klassen und das Parsen mit JDOM ist/sind korrekt? Bitte beachten Sie: Falsche Antworten führen zu Punktabzug.

☐ Die Document-Klasse lädt das komplette XML-Dokument als Baum in den Speicher.

☐ Das Selektieren des Wurzelknotens beim Parsen von XML-Dokumenten ist nicht zwingend erforderlich.

☐ Keine der obigen Aussagen ist korrekt.