PART A

Candidates should answer **ALL** of Question 1 in Part A.

Question 1

(a) Normalisation can be thought of as an attempt to reduce information duplication in a database. What is the ONE primary reason why this is desirable? (note: several of these reasons are valid)

Choose ONE option [4]

i. Database storage was expensive when the theory was formalised. Even a reduction of 10% on storage requirements could save thousands of pounds.
ii. Duplicate information can get out of sync, leading to logical inconsistencies in the data
iii. Duplicate information takes more time to input. Adding extra structure through normalisation speeds data entry.
iv. Redundant data represents a significant security threat, since it provides patterns in your database that can be exploited by hackers

ii

(b) I want to query a database table of books and join it to a table of authors. Where the book is anonymous, I want the author field to have the value `NULL`. What type of join do I need?

Select ONE correct option [4]

i. LEFT JOIN
ii. INNER JOIN
iii. CROSS JOIN
iv. INNER JOIN and special WHERE clause

i

(c) Look at the following sequence of commands and then choose the statement from the list that should replace XXXXXXX

```
START TRANSACTION;
SELECT stockLevel FROM Stock WHERE ID=23948267;
```

```
UPDATE Stock SET stockLevel=stockLevel-2 WHERE ID=23948267;
INSERT INTO Orders (productID, quantity) VALUES (23948267, 2) WHERE ID=292484
XXXXXXX
```

Choose ONE option. [4]

i. `GRANT ALL ON * TO 'Stock' WITH GRANT OPTION;`

ii. `UPDATE Stock SET stockLevel=2;`

iii. `END TRANSACTION;`

iv. `COMMIT;`

v. `UPDATE Stock SET stockLevel=stockLevel+2 WHERE ID=23948267;`

iv

(d) What does the following MongoDB call do?

```
db.books.find({
  title: /*man/,
  author: /*man/,
  year: 1934
});
```

Choose ONE option. [4]

i. Sets the year to 1934 for books with a title or author called man

ii. Finds all books with title and author called man

iii. Finds all books with title and author ending in man

iv. Finds all books with title and author called man with a year of 1934

v. Finds all books with title and author ending in man with a year of 1934

vi. Adds a new record to books with title ”*man”, author ”*man” and year 1934

iv

(e) Why are URLs important in Linked Data and other Semantic Web technologies? Select ALL correct statements. [4]

i. A URL is necessarily unique, and so can be used like a Primary Key

ii. A URL is shareable, so anyone can refer to it when they want to show they mean the same thing

iii. A URL can encode meaning in its path that can be parsed into useful information

iv. HTTP requests for a URL can be 'dereferenced' – they can have responses that provide useful information about the thing the URL represents

v. A URL is permanent and reliable, providing important guarantees for any data

i, ii, iv

(f) Given the XML fragment below, what does the XPath expression `//note/title` refer to?

```
<msContents>
  <textLang mainLang="en">English</textLang>
  <msItem n="1" xml:id="Christ_Church_MS_473-item1">
    <locus> fol. 1{26</locus>
    <rubric>Certain humble Petitions which are in most humble manner...
    </rubric>
    <incipit>1. That there may be a veiwe taken of all the
                markett Townes</incipit>
    <note>Printed by John Strype, <title>Annals of the Reformation</title>
      (London, 1709), 3:Appendix (68{81), working from what is now BL,
      MS Lansdowne 119. Strype, <title>Annals</title>, 3:221 attributes
      both the Petition and the related 'Supplication', which follows in
      our codex, to the Puritan divine,
      <persName key="person_228371281"
                role="aut">Thomas Sampson (d. 1589)</persName>.
    </note>
  </msItem>
  <msItem n="2" xml:id="Christ_Church_MS_473-item2">
    <locus> fol. 28{34<hi rend="superscript">v</hi></locus>
    <author key="person_228371281"/>
    <incipit defective="true">and painfull preachinge pastors
                             resident among us</incipit>
    <explicit>obey you accordinge to his blessed will thorowly
    </explicit>
    <note>The complete text, titled <title>A Supplication made
      in the Name of certain true subjects...</title> and opening
      ‘In most humble wise complaining, we which are Thousands
      of the poor untaught People of England...’, is printed by Strype,
      <title>Annals</title>, 3:222{27. Our copy has lost the title
      and first 30 lines, as printed by Strype, equating presumably
      to one recto and verso. </note>
  </msItem>
</msContents>
```

Select ONE correct statement. [4]

i. The four `title` elements that appear as direct child nodes of `note` elements
ii. Nothing, since no child of the root element is a `note` elements
iii. The first `title` element that in a `note`
iv. A `title` element, but there are none in this fragment
v. The two `note` elements that have at least one child element that is a `title`

i

(g) Which of the following are true statements about MapReduce?

i. The Map phase is carried out on local data
ii. The Reduce phase is carried out on local data
iii. The Map phase produces data that can be distributed based on key
iv. The Reduce phase produces data that can be distributed based on key
v. The input data for MapReduce systems can be distributed
vi. Reducer operations can be distributed
vii. MapReduce reduces the amount of processing needed
viii. MapReduce makes it easier to run the processing in parallel

Select ALL correct statements. [4]

i, iii, v, vi, vii

(h) Consider the following code:

From the music ontology:

```
mo:listened a owl:ObjectProperty
            dfs:comment "Relates agents to the
                         performances they were listening in" ;
            rdfs:domain foaf:Agent ;
            rdfs:range mo:Performance ;
            owl:inverseOf mo:listener .
```

And from some published data:

```
orcid:0000-0003-4151-0499 a foaf:Agent ;
            mo:listened <http://data.carnegiehall.org/events/46018> .

<http://data.carnegihall.org/events/46018>
    dct:date 2008-03-28T19:30:00 ^^xsd:dateTime .
```

Given these statements, which of the following MUST be true?

i. `orcid:0000-0003-4151-0499 a mo:Performance .`
ii. `<http://data.carnegiehall.org/events/46018> a foaf:Agent .`
iii. `<http://data.carnegiehall.org/events/46018> a mo:Performance .`
iv. `orcid:0000-0003-4151-0499 dct:date "2008-03-28T19:30:00"8sd:dateTime .`
v. `<http://data.carnegiehall.org/events/46018> mo:listener orcid:0000-0003-4151-0499 .`

Select ALL correct statements. [4]

iii, v

(i) A retrieval algorithm has a fairly constant 20% precision across all result sets. If my corpus contains 15,030,482 documents of which 225,030 are relevant, which of the following statements are likely to be true?

i. A result set of 80 documents will have about 32 correct matches.
ii. A result set of 80 documents will have about 16 correct matches.
iii. There will be at most about 45,000 correct matches returned.
iv. The algorithm does over 10 times better than random.
v. The algorithm does over 100 times better than random.

Select ALL correct statements. [4]

ii, iv

(j) Which of the following statements is true of **Copyleft**

i. Copyleft licences are described as **permissive**
ii. Copyleft licences are not **permissive**
iii. Copyleft licences allow derivative works freedom to choose their own licensing systems
iv. Copyleft licences restrict the licensing of derivative works to ensure that they are also Copyleft
v. The GNU Public Licence (GPL) is an example of Copyleft
vi. The MIT licence is an example of Copyleft

Select ALL correct statements. [4]

ii, iv, v

PART B

Candidates should answer any **TWO** questions from Part B.

Question 2

The following text describes a television series:

> Doctor Who was first broadcast in 1963, with William Hartnell playing what we now call the ‘First Doctor’. From the start, the Doctor was accompanied by companions. The first companions were his granddaughter, Susan Foreman, and two of her schoolteachers, Ian Chesterton and Barbara Wright. Susan was played by Carole Ann Ford.
>
> In 1966, an episode was broadcast in which the Doctor collapses from exhaustion. His body regenerates into the ‘Second Doctor‘, played by Patrick Troughton.
>
> In total, between 1963 and the present, there have been 13 numbered incarnations of the Doctor, and one extra incarnation called ‘The War Doctor‘, along with dozens of companions. Although most companions appear exclusively with one Doctor, some overlap with several, and some take guest appearances in episodes with later Doctors.

(a) We wish to design a simple database about this series, modelling Doctors, their companions, who played them and when they were broadcast. List the tables needed, indicating all keys. [6]

- ***Doctor**: designation[PK], actor, firstAppearance, lastAppearance*
- ***Companion**: Name[PK], actor*
- *either **CompanionDoctor**: companion[FK], doctor[FK][PK]*
- *or **CompanionAppearance**: companion[FK], appearance[PK]*

*Accept sensible alternatives (including DoctorAppearance as a table instead of first and last) **+3 marks** for tables with good coverage of data **+3 marks** for keys*

(b) Give a MySQL command for creating ONE of these tables [3]

```
CREATE TABLE Doctor (
  name VARCHAR(25) PRIMARY KEY,
  actor VARCHAR(50),
  firstAppearance DATE,
  lastAppearance DATE
)
```

__+1 mark__ for correct and complete fields, __+1 mark__ for main command syntax, __+1 mark__ for correct key syntax and spec, along with any constraints.

(c) Are your tables in 2NF? How can you tell? [3]

If the tables were well defined in the first place, the answer should be YES (but only give credit if correct)__1 mark__ . The tables are in 1NF (because their field values aren't themselves tables) __1 mark__ and no attribute or group of attributes that is not part of the candidate key is functionally dependent on a part of the candidate key __1 mark__.

(d) Give MySQL commands to answer the following queries in your schema:

i. Who played the Doctor whose companion was Amy Pond? [2]

ii. Was (companion) Peri featured before Leela? (or was Peri's Doctor before Leela's?) [4]

iii. Which incarnation of the Doctor had the most companions? [3]

This will depend on the table designs, but:

```
SELECT  Actor
  FROM  CompanionDoctor INNER JOIN Doctor ON doctor=designation
  WHERE companion = "Amy Pond";

SELECT dp < dl
  FROM
  (SELECT firstAppearance AS dl
     FROM CompanionDoctor INNER JOIN Doctor ON doctor=Designation
    WHERE Companion LIKE "%Leela%" ) AS LeelaFirst,
  (SELECT lastAppearance AS dp
     FROM CompanionDoctor INNER JOIN Doctor ON doctor=Designation
    WHERE Companion LIKE "%Peri%" ) AS PeriLast;

SELECT   Doctor
  FROM   CompanionDoctor
GROUP BY Doctor
```

```
ORDER BY COUNT(*) DESC
   LIMIT 1;
```

(e) Here is an extract from the DBpedia entry for the First Doctor:

```
dbr:First_Doctor rdfs:label       "First Doctor"@en ;
                 dbp:periodEnd    "1966-10-29"^^xsd:date ;
                 dbp:periodStart "1963-11-23"^^xsd:date ;
                 dbp:companions  "Ben Jackson"@en ,
                                 "Vicki"@en ,
                                 "Sara Kingdom"@en ,
                                 "Steven Taylor"@en ,
                                 "Susan Foreman"@en ,
                                 "Polly"@en ,
                                 "Ian Chesterton"@en ,
                                 "Barbara Wright"@en ,
                                 "Dodo Chaplet"@en ,
                                 "Katarina"@en ;
                 dct:subject     dbc:Doctor_Who_Doctors ;
                 dbp:next        dbr:Second_Doctor .
```

i. What serialization language is this? [1]

ii. How many triples are encoded here? [1]

iii. What can your database schema do that this approach can't? [2]

iv. How would you fix that problem? [2]

v. Fix the following SPARQL query, which should look for any incarnation of the Doctor who had Amy Pond as a companion

```
SELECT dbr:doctor
WHERE {
  dbr:First_Doctor dbp:next+ ?doctor .
  dbr:doctor dbp:companion "Leela" .
}
```

[3]

.

i. Turtle (ttl)

ii. 15

iii. Store information about the companions (e.g. actor)

iv. Give companion a URL

v.
```
SELECT ?doctor
WHERE {
  dbr:First_Doctor dbp:next+ ?doctor .
  ?doctor dbp:companion "Amy Pond" .
}
```

Question 3

(a) Consider the following XML fragment

```
<castList xmlns="http://www.tei-c.org/ns/1.0">
  <castGroup>
    <castGroup>
      <head>four lovers</head>
      <castItem xml:id="Hermia_MND">
        <role>
          <name>Hermia</name>
        </role>
      </castItem>
      <castItem xml:id="Lysander_MND">
        <role>
          <name>Lysander</name>
        </role>
      </castItem>
      <castItem xml:id="Helena_MND">
        <role>
          <name>Helena</name>
        </role>
      </castItem>
      <castItem xml:id="Demetrius_MND">
        <role>
          <name>Demetrius</name>
        </role>
      </castItem>
    </castGroup>
  </castGroup>
  <castGroup>
    <castItem xml:id="Theseus_MND">
      <role>
        <name>Theseus</name>
      </role>
      <roleDesc>duke of Athens</roleDesc>
    </castItem>
  </castGroup>
```

i. This fragment is unbalanced. What is missing [1]

ii. What format is this? [1]

iii. What attributes are used in this fragment? [2]

i. Missing close tag for the `castList` element [1]

ii. TEI [1] (allow only 1/2 for XML)

iii. `xmlns` and `xml:id` (1 each)

(b) Here is an edited extract from the associated xsd file

```
<xs:element name="castGroup">
  <xs:annotation>
    <xs:documentation>(cast list grouping) groups one or more individual castI
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="ns1:model.global"/>
        <xs:group ref="ns1:model.headLike"/>
      </xs:choice>
      <xs:sequence maxOccurs="unbounded">
        <xs:choice>
          <xs:element ref="ns1:castItem"/>
          <xs:element ref="ns1:castGroup"/>
          <xs:element ref="ns1:roleDesc"/>
        </xs:choice>
      </xs:sequence>
    <xs:attributeGroup ref="ns1:att.global.attributes"/>
  </xs:complexType>
</xs:element>
```

i. What is this file and what does it it do? [2]

ii. The `castGroup` element as defined here begins with `model.global` or `model.headlike` elements. Neither happens the first time this element occurs in the fragment above. Does this mean the document is invalid? Explain your answer. [3]

iii. Do the `castGroup` elements in the fragment follow this definition correctly? Explain your answer. [3]

i. It is an XML Schema Definition [1] and formally describes the syntax for elements in the encoding [1]

ii. *No, this does not invalidate the document [1] because these elements are allowed to occur 0 or more times (`minOccurs='0'`) [1] here, they occur 0 times [1]*

iii. *Yes, the elements are valid [1], but only if head is in `model.headLike`[1]. `castGroup` only contains `castGroup` and `castItem` elements as direct children, except for the one instance of `head` (with content 'four lovers') [1]*

(c) Consider this extract of an xsl file associated with this fragment:

```
<xsl:template match="castItem">
  <div>
    <xsl:apply-templates match="node()|@*"/>
  </div>
</xsl:template>
<xsl:template match="role/name">
  <dt>
    <xsl:value-of select="."/>
  </dt>
</xsl:template>
<xsl:template match="roleDesc">
  <dd>
    <xsl:value-of select="."/>
  </dd>
</xsl:template>
```

i. What is this file and what is it for? [2]

ii. This fragment won't work. What is missing from the `match` attributes? [2]

iii. What will be the format and content of the output of applying this file (when it works)? [3]

i. *XML Stylesheet Language Transformation [1], which gives templates for transforming an XML file into something else [1]*

ii. *There are no namespaces [1]. A namespace prefix would need to be declared, and then applied (e.g. `match="tei:castItem"`)[1]*

iii. *At least based on the tags used, it looks like the output will be HTML or XHTML,[1] with each character in the cast a definition in a definition list (`dl`) [2] (Note that the `dl` element is unusual for a list in that it allows divs to enclose definitions)*

(d) You have been tasked with representing the same data, but using the relational model

i. What tables would you use to represent this cast information? [6]

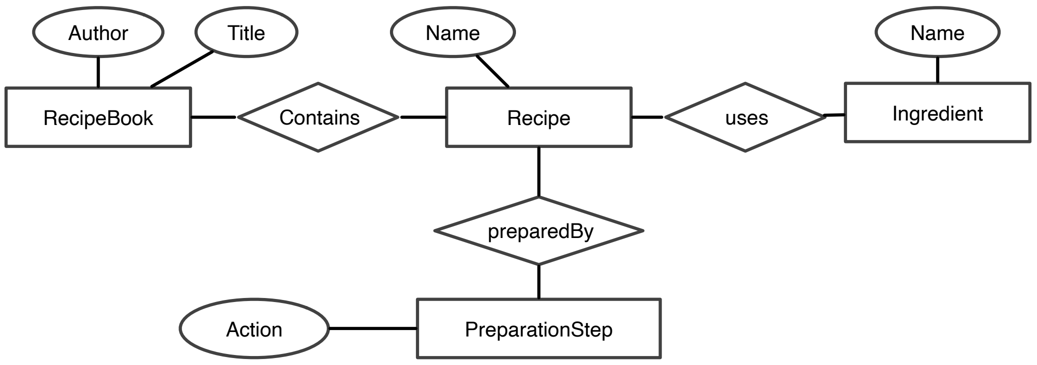
ii. Which works better here, the relational model or the XML implementation? Why? [5]

i. For this data, I would have [CastGroup: ID (INT) (PK), head (VARCHAR(80))], [CastItem: XMLID (VARCHAR(50)) (PK), roleName (VARCHAR(50)), roleDesc (VARCHAR(80))], [GroupItems: GroupID(FK), ItemID(FK)(PK)], [GroupGroups: GroupID(FK), subGroupID(FK)(PK)] and, optionally, [ListGroups: ListID (INT), GroupID(FK) (PK)]

ii. Any sensible points rewarded at 1 mark each. Observation and evaluation can count for 2, e.g. castGroup models a nested hierarchy [1], which is better handled by a tree structure such as XML [1]

Question 4

A friend has sketched their idea for a recipe database in an E/R diagram. This database will power a website that will allow browsing by ingredient, or by recipe books and their authors, as well as just by recipe name.



(a) Do you think there's anything missing from this model that the database might need? If so, what? [4]

(b) What is the cardinality of the three relationships? [3]

(c) Give MySQL-compatible CREATE TABLE commands for TWO of the tables that you would use to implement the database (including any corrections from (a) above). Include any appropriate key constraints. [10]

The examples below are illustrative. Each table can get up to 6 marks (capped at 10 overall), with credit for: Correct and complete fields identified [1]; Correct and complete PKs [1]; correct and complete FKs [1]; Good data type choice [1]; Correct syntax [1]; (bonus) Appropriate use of FK constraint with ON DELETE/UPDATE clauses [1]

```
CREATE TABLE RecipeBook (
  title VARCHAR(100),
  author VARCHAR(100),
  PRIMARY KEY (title, author)
)
CREATE TABLE Recipe (
  ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100)
)
CREATE TABLE RecipeReference (
  recipeID INT,
  title VARCHAR(100),
  author VARCHAR(100),
  PRIMARY KEY (title, author, recipeID),
  FOREIGN KEY (title, author)
    REFERENCES RecipeBook(title, author)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (recipeID)
    REFERENCES Recipe(ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
CREATE TABLE RecipeIngredient (
  ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  recipeID INT,
  ingredient VARCHAR(100),
  quantity VARCHAR 20,
  FOREIGN KEY (recipeID)
    REFERENCES Recipe(ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (ingredient)
    REFERENCES Ingredient(name)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
)
CRETAE TABLE Ingredient (
  name VARCHAR(40) PRIMARY KEY
)
CREATE TABLE PreparationStep (
  recipeID INT,
  step     TINYINT UNSIGNED,
  action   VARCHAR(255),
  PRIMARY KEY (recipeId, step),
  FOREIGN KEY (recipeID)
    REFERENCES Recipe(ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
)
```

Also consider making `recipeIngredient` link to the preparation step where it is used rather than the recipe – it would still be listable for the recipe, but *it now would be clear where it is used.*

(d) Give a single MySQL query to find all the recipe books which never use butter. [4]

There are multiple ways to do this, and it will depend on table implementation. A basic query with good general syntax [1], good join logic and syntax[1], more complex syntax (WHERE and/or subselects) is also correct[1], does the job correctly[1].

```
SELECT title, author
  FROM RecipeBook NATURAL JOIN RecipeReference
                  LEFT JOIN Recipe ON (RecipeID=ID)
 WHERE NOT EXISTS (SELECT *
                    FROM  RecipeIngredient ri
                    WHERE ri.RecipeID=Recipe.ID
                          AND ingredient="butter");
  GROUP BY zoo;
```

(e) Give a single MySQL query to find the average number of steps in the recipe book called ”Mushrooms” by ”John Cage”. [4]

```
SELECT AVG(steps)
  FROM (SELECT   COUNT(*) AS steps
          FROM   PreparationStep ps
                 LEFT JOIN RecipeReference rr ON (rr.RecipeID = ps.RecipeID)
         WHERE   title="Mushrooms" AND author="John Cage"
        GROUP BY rr.recipeID) AS recipeSteps);
```

Core syntax [1], logic and joins [1], complex syntax [1], query works[1]

(f) Consider ONE of the following technologies and indicate whether you think they would be more suitable for this database. Explain your answer.

- Document database (MongoDB)
- XML document resource or database
- Linked Data resource or triplestore

[5]

Credit per reasonable, substantial point. This data can be seen as quite hierarchical, which would favour MongoDB or XML, but it also has inter-connections *that make more like a graph, favouring LD. Extending the model beyond the minimum given here (such as nutritional info) would seem to favour LD. Some of this discussion will depend on the modelling of the tables in earlier sections (e.g. if a recipe only appears in one book, the data looks more tree-like).*

END OF PAPER

.