UNIVERSITY OF LONDON

CM2040

BSc EXAMINATION

COMPUTER SCIENCE

Databases, Networks and the Web

Release date: Wednesday 16 September 2020: 12.00 midday British Summer Time

Time allowed: 24 hours to submit

Submission date: Thursday 17 September 2020: 12.00 midday British Summer Time

INSTRUCTIONS TO CANDIDATES:

Part A of this assessment consists of a set of 10 Multiple Choice Questions (MCQs) which you will take separately from this paper. You should attempt to answer **ALL** the questions in Part A. The maximum mark for Part A is **40**.

Part A will be completed online on the VLE. You may choose to access the MCQs at any time following the release of the paper, but once you have accessed the MCQs you must submit your answers before the deadline or within **4 hours** of starting, whichever occurs first. Candidates only have **ONE** attempt at Part A.

Part B of this assessment is an online assessment to be completed within the same 24-hour window as Part A. We anticipate that approximately **1 hour** is sufficient for you to answer Part B. Candidates must answer **TWO** out of the **THREE** questions in Part B. The maximum mark for Part B is **60**.

Calculators are not permitted in this examination. Credit will only be given if all workings are shown.

You should complete **Part B** of this paper and submit your answers as **one document,** if possible, in Microsoft Word or a PDF to the appropriate area on the VLE. You are permitted to upload 30 documents. However, we advise you to upload as few documents as possible. Each file uploaded must be accompanied by a coversheet containing your candidate number. In addition, your answers must have your **candidate number** written clearly at the top before you upload your work. Do not write your name anywhere in your answers.

PART B

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

Question 1

Consider the home insurance database below, where each line shows a table with its properties, *TableName (column1, column2, ...)*, and the primary and foreign keys are underlined. The first column in each table is a primary key and other underlined column names are foreign keys referring to the column with the same name in another table. As an example, 'report_number' in the 'Happened' table is a foreign key referring to the primary key 'report_number' in the 'Loss_event' table.

> *Customer (<u>customer_id</u>, name, address)*
> *Home (<u>home_id</u>, address, year)*
> *Loss_event (<u>report_number</u>, year, type)*
> *Owns (<u>id</u>, <u>customer_id</u>, <u>home_id</u>)*
> *Happened (<u>id</u>, <u>report_number</u>, <u>home_id</u>, <u>customer_id</u>, damage_amount)*

You are a freelance backend developer responsible for implementing a database for an insurance company, for which you were given the information above.

(a) Describe what a junction or bridge table is in the context of database design. [4]

(b) When designing a database, which issues can arise when using many-to-many associations? Use an example and illustrations to clarify your answer. How can a junction or bridge table resolve the issues? [8]

(c) Which table in the home insurance database fulfils the junction table role? Justify your answer. [2]

(d) Draw an Entity-Relationship diagram for the insurance database. Your diagram should include entities and relationships between entities and suitable association types. [9]

(e) Write the SQL code to create the Owns table. Make sure it includes suitable properties to represent the concept that a Customer owns a Home. [7]

Question 2

Consider the insurance database below, where each line shows a table with its properties, *TableName (column1, column2, ...)*, and the primary and foreign keys are underlined. The first column in each table is a primary key and the other underlined column names are foreign keys referring to the column with the same name in another table. As an example, 'report_number' in the 'Happened' table is a foreign key referring to the primary key 'report_number' in the 'Loss_event' table.

Customer (<u>customer_id</u>, name, address)
Home (<u>home_id</u>, address, year)
Loss_event (<u>report_number</u>, year, type)
Owns (<u>id</u>, <u>customer_id</u>, <u>home_id</u>)
Happened (<u>id</u>, <u>report_number</u>, <u>home_id</u>, <u>customer_id</u>, damage_amount)

(a) Write a SQL statement that finds the total number of loss events in and after 2020. [4]

(b) State the purpose of a SQL JOIN statement and explain how it works. [4]

(c) Explain the difference between SQL JOIN, SQL RIGHT JOIN, and SQL LEFT JOIN. [6]

(d) Considering the database specification above, write a SQL JOIN statement that finds the total amount of damage associated with events involving a customer named John Smith. [8]

(e) Write a SQL statement to find all types of loss events where a customer named 'John Smith' was involved. [8]

Question 3

You are a full-stack developer building an online e-commerce web application for a company selling jewellery online. You have written a middleware script with a route that adds collected form-data related to a new piece of jewellery to the database as follows:

```
app.post('/jewelleryadded', function (req,res) {
  let squery = "INSERT INTO jewellery (category, name, price) VALUES (?,?)";
  console.log (squery);
  let rec = [req.body.category, req.body.name, req.body.price];
  db.query(squery, rec, (err, result) => {
  if (err) {
    return console.error(err.message);
  }
  else
    res.send('Data related to one piece of jewellery added to database');
  });
});
```

The web server code named 'index.js' is shown below:

```
1.  var express = require ('express')
2.  var bodyParser= require ('body-parser')
3.  var mysql = require ('mysql')
4.  const app = express()
5.  const port = 8000
6.  const db = mysql.createConnection ({
7.             host: 'localhost',
8.             user: 'root',
9.             password: 'password',
10.            database: 'OnlineJewelleryShop'
11. });
12. db.connect((err) => {
13.   if (err) {
14.     throw err;
15.   }
16.   console.log('Connected to database');
17. });
18. global.db = db;
19. app.use(bodyParser.urlencoded({ extended: true }))
20. require('./routes/main')(app);
21. app.set('views',__dirname + '/views');
22. app.set('view engine', 'ejs');
23. app.engine('html', require('ejs').renderFile);
24. app.listen(port, () => {
25.    console.log(`Example app listening on port ${port}!`)

26. })
```

(a) Which line of code would you change to change the console output of the web server (web server prompt)? [1]

(b) Which line of code starts to add a 'route' to your web application? How do you know it is this line? [2]

(c) Why do you need to import body-parser (line 14) in your Node.js web server code? [2]

(d) Which line of code accesses data collected from a POST request? Which form-fields does it access? [4]

(e) Describe what templating is in the context of web application development. [4]

(f) Write a template file using the EJS embedded javascript templating library. It should display all jewellery items passed as a parameter from the middleware to the template file. The template file should generate a complete HTML page, it should have a title using a suitable HTML tag, the incoming variable is called availableItems. Each jewellery item stored in the database has three properties of category, name, and price. [6]

(g) Write a piece of middleware code, with a route named 'display_all_rings' to access the template file written in part (f) and retrieve and display all jewellery items in the 'ring' category stored in the database. Your code should query the database to get all rings. Your code should handle the error condition where the database query fails. Your code should render a template file to display all retrieved items from the database. [7]

(h) You have been assigned a new project to develop a dynamic web application for a hospital records system using Node.js and MySQL. What are the steps you would follow as a full-stack developer for the development of this web application? Name at least two main steps for the design and two main steps for the implementation of the web application considering what you have learned in this module. [4]

END OF PAPER