The University of Hong Kong

Faculty of Engineering
Department of Computer Science

COMP3322B Modern Technologies on World Wide Web

Date: May 15, 2021 Time: 2:30 pm - 4:30 pm

Instructions

- **This is an open-book examination. Candidates are permitted to bring to the examination hardcopies or softcopies of all lecture slides, and workshop worksheets. They can use the built-in developer tools in the browsers to assist their coding.**
- **Candidates are allowed to search the Internet for information during the examination. However, crowdsourcing from group messages, online forums or social media, etc. is strictly forbidden.**
- **Total of 100 points. This exam consists of 7 questions. Candidates are required to answer all questions.**
- **If you are not clear about what a question is asking, be sure to write down any assumptions you have made in answering the question.**
- **To answer the questions, you can**
    - **use a text editor to type in the answers – this is the preferred option.**
    - **type the answers in a Microsoft Word document. When inserting program code into the Word document, insert the code fragment as ASCII plain text; don't insert the code fragment as an image.**
- **Don't write your answers on paper.**
- **Put your University Student Number at the beginning of the answer script, followed by the Course code of this examination. Enter the name and type of your calculator, if any, after the Course code.**
- **Before submitting the answer file to the OLEX system, convert the Word file or text file to a single PDF file.**
- **Please make sure that the submitted PDF file:**
    - **It includes all your answers, and the answers are arranged in the question order.**

Question 1. Short Questions (18%)

a) (4) Describe how a web client knows whether it can cache a web object and for how long after it has received the object from the web server.

b) (5) Explain two ways to improve the accessibility of an image embedded in a Web page for users with different abilities.

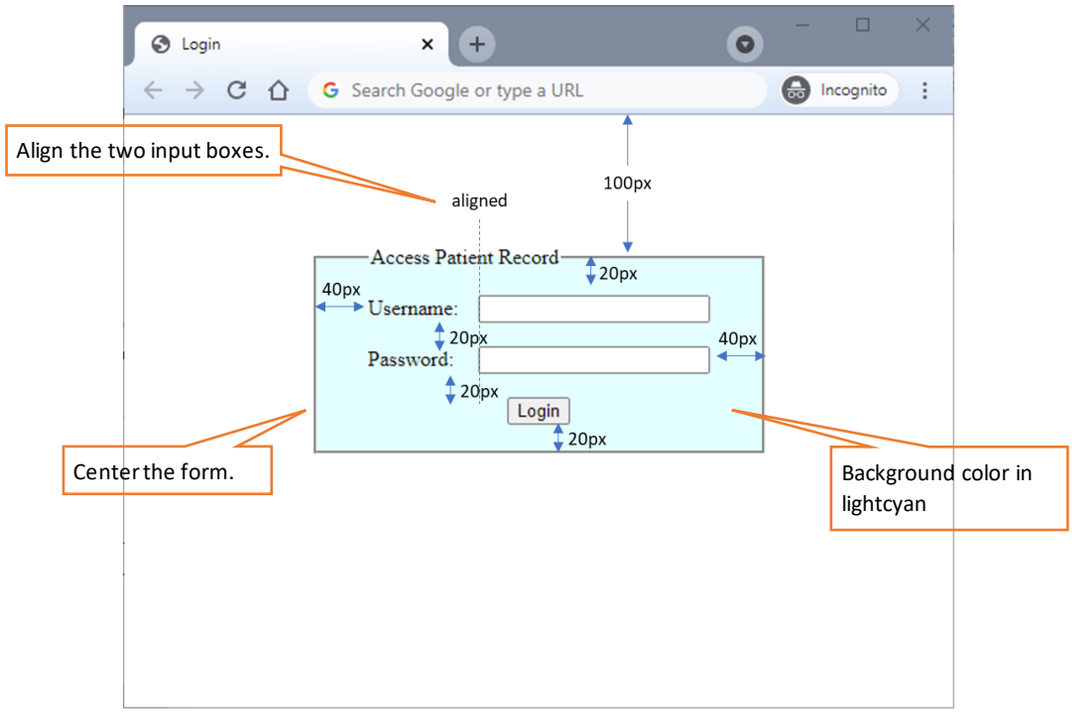
c) (4) HTML forms can be submitted to the server by the GET or POST method. Explain the differences between the two methods with regards to the (i) URL, (ii) headers, and (iii) message body.

d) (5) Consider that a user wants to retrieve the Mozilla Developer Network home page (`https://developer.mozilla.org/en-US/`) but his computer does not have the IP address of the Web server. Describe the process of how his computer obtains the IP address of the Web server. Assume that the local DNS server has not cached the address.

Question 2. CSS (10%)

Below are the HTML code of a page and the image of that page rendered on the Chrome Web browser with the CSS styling. Create the CSS styling rules according to the hints and information given along with the displayed image. You CANNOT make changes to the body part of this HTML code, i.e., no class or id attributes can be added to the code. You CANNOT make use of any JavaScript code too.

```
<body>
  <form action="login" method="post">
    <fieldset name="logininfo">
      <legend>Access Patient Record</legend>
      <label for="username">Username:</label>
      <input type="text" name="username" id="username"><br>
      <label for="password">Password:</label>
      <input type="password" name="password" id="password"><br>
      <input type="submit" name="login" value="Login">
    </fieldset>
  </form>
</body>
```



Question 3. Canvas (12%)

Consider a canvas HTML element is declared in the body part of a web page by the following statement.

```
<canvas id="Canvas1" width="640" height="400" style="border:1px solid #c3c3c3;">
  Your browser does not support the canvas element.
</canvas>
```

We are going to design a JavaScript function named run() that uses this canvas element like a photo slider. Upon execution, this function performs the following tasks.

1. The function first uses the fetch() AJAX call to retrieve a json file named photo.json from the /images directory. This file contains all the filenames of the photos that the program is going to render to the canvas. Here is an example of the photo.json file.
   `[ "image1.png", "image2.png", "image3.jpg" ]`
   The function assumes all the photos and the photo.json file are stored in the /images directory.
2. Given the list of photos, the function displays the photo one by one, each for 8 seconds. After displaying the last photo, the program goes back to the start of the list and loops again.
3. Each photo is being displayed and aligned in the middle of the canvas. When the photo is larger than the canvas's height or width, the program scales the photo to a suitable size for displaying it in the middle of the canvas.

Here are some example screenshots of the output of this program.

image1.png (4040 x 2693)

image2.png (418 x 700)

image3.jpg (298 x 298)

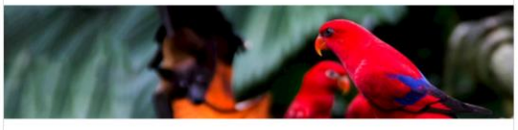


image4.jpg (1800 x 400)

Question 4. PHP (12%)

Write a PHP file named visit.php that uses a session to keep track of how many times a client has visited the website on `"www.code.co/mysite/"` and its sub-directories since the browser started. You are going to count how many times the client has visited a specific path of that website. For example, there is a counter

for the path "/mysite/" and another counter for the path "/mysite/about/". This fragment of code will be placed in all .php files under this website by using the "require_one" statement. If two .php files are under the same prefix path, e.g., "/mysite/service/get.php" and "/mysite/service/set.php", they should share the same counter.

Question 5. Express and MongoDB (24%)

Write an express.js program called **index.js** that pulls out data from a MongoDB database (called **course-data**) which stores the examination results. The examination results are grouped in a collection called ***gradebook*** which has the following structure. You can consider the column headers as the field names of the collection. (Note: for brevity, we do not show the '_id" field here.) Assume the MongoDB server is running on localhost with the default port number 27017.

stdName	stdNumber	course	score
XXXX Lee	3015111111	COMP2119	54
XXXX Lee	3015111111	COMP3322	68
XXXX Lee	3015111111	COMP3297	65
XXXX Lee	3015111111	COMP3230	60
XXXX Lee	3015111111	COMP3278	72
YYYY Wong	3015222222	COMP3322	80
YYYY Wong	3015222222	COMP3297	76
YYYY Wong	3015222222	COMP3230	76
YYYY Wong	3015222222	COMP3278	69
ZZZZ Ko	3015333333	COMP2119	72
ZZZZ Ko	3015333333	COMP3297	68
ZZZZ Ko	3015333333	COMP3230	65
ZZZZ Ko	3015333333	COMP3278	75
PPPP Ho	3015444444	COMP2119	72
PPPP Ho	3015444444	COMP2120	70
PPPP Ho	3015444444	COMP3322	85
PPPP Ho	3015444444	COMP3297	78
PPPP Ho	3015444444	COMP3230	75
PPPP Ho	3015444444	COMP3278	81

a) (6) Write the code in your express app to (i) set up a connection to MongoDB using Mongoose for accessing the data, and (ii) set up a schema and a model for accessing the gradebook data. Assume the program has executed the following statements before calling your code.

```
const express = require('express')
const app = express();
```

Also, assume that the express app is listening to port 8000 for all incoming HTTP requests.

b) (8) Write an endpoint to handle the GET request for retrieving the examination scores of all students on a course. The returned data should be a JSON string.

To retrieve the examination scores of the course COMP3322, the client sends the GET request to `/COMP3322/getscore`. To retrieve the examination scores of the course COMP2119, the client sends the GET request to `/COMP2119/getscore`.

This endpoint performs the following tasks:

- Only response to GET requests.
- Connect to the database to retrieve the stdName, stdNumber, and score of all documents that match the requested course code.
- If no matched document is returned, send the JSON string `"[]"` to the client.
- If there are matched documents, send all returned results as a single JSON string to the client. Please make sure that the returned results only contain the stdName, stdNumber, and score fields. For example, the following is the returned result for `/COMP2120/getscore`.
  `[{"stdName":"PPPP Ho","stdNumber":"3015444444","score":70}]`
- If encountering error in retrieving data, respond to the client with status code 500.

c) (10) Write another endpoint to handle the GET request to the server for retrieving examination scores for a student. This endpoint supports two different path formats. A student with student number 3015222222 can use this path `'/student/3015222222'` to retrieve the examination scores of all his enrolled courses. Or he uses this path `'/student/3015222222/COMP3297'` to retrieve the examination score of the course COMP3297. The returned data should be a JSON string.

This endpoint performs the following tasks:

- Only response to GET requests.
- Only response to the matched path patterns, e.g., `/student/3015222222`, `/student/3015333333/`, `/student/3015444444/COMP3230`, `/student/3015444444/COMP3297/`, etc. You can assume that the client always passes a correct student number and a course code in the requests.
- If not receiving a GET request or the requested path pattern does not match, don't handle the request and pass the request to the built-in express error handling middleware.
- If encountering an error in retrieving data from the database, respond to the client with status code 500.
- If receiving a request for retrieving all examination scores for a student, connect to the database to retrieve the course and score fields of all matched documents. For example, the following is the returned result for `'/student/3015222222'`.
  `[{"course":"COMP3322","score":80},{"course":"COMP3297","score":76},{"course":"COMP3230","score":76},{"course":"COMP3278","score":69}]`
- If receiving a request for retrieving an individual examination score for a student, connect to the database to retrieve the course and score fields of the matched document. For example, the following is the returned result for `'/student/3015222222/COMP3322'`.
  `[{"course":"COMP3322","score":80}]`
- If no matched document is returned, send the JSON string `"[]"` to the client.

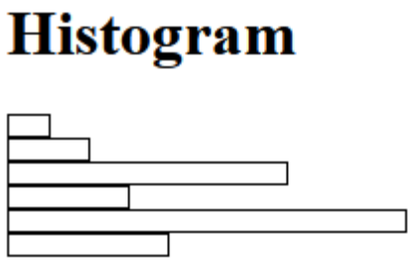
Question 6. Express and Pug (12%)

Write a pug template file named histogram.pug. This pug template file accepts one input object, which is an array of integer. The template will output a block for each integer k in the array with the following CSS setting: border = 1px solid black, height = 10px, and width = k * 20px .

For example, with the input object dataset = [1, 2, 7, 3, 10, 4], by using the following express statement -

```
res.render('histogram', {data: dataset}),
```

the following HTML response will be sent to the browser.




```
<!DOCTYPE html>
<html>
<head>
  <title>Histogram</title>
</head>
<body>
  <h1>Histogram</h1>
  <div id="container">
    <div style="border:1px solid black;height:10px;width:20px;"></div>
    <div style="border:1px solid black;height:10px;width:40px;"></div>
    <div style="border:1px solid black;height:10px;width:140px;"></div>
    <div style="border:1px solid black;height:10px;width:60px;"></div>
    <div style="border:1px solid black;height:10px;width:200px;"></div>
    <div style="border:1px solid black;height:10px;width:80px;"></div>
  </div>
</body>
</html>
```

Question 7. React (12%)

Design a React component named Dclock to render a digital clock, which displays the time within an <h2> tag, i.e., <h2>17:53:7</h2>. When being displayed, it would look as follows:

17:53:7

Note:

i. The clock will update the time periodically, i.e. in every second.
ii. You can use a JavaScript Date object to get the current time.
iii. We are expected to use the following React call to add the component to our application.

```
ReactDOM.render(<Dclock />, document.getElementById("myroot"));
```

***** END OF PAPER *****