CSE 154 Exam 2 Answer Booklet

Name: _____

UWNet ID: _____**@uw.edu**

TA (or section): _____

Rules:

- You have 60 minutes to complete this exam.
- **There is another "Exam 2 Problem Reference Booklet" with specifications and provided code for Problems 2, 3, and 4. All answers must be written in *this* exam booklet.**
- You will receive a deduction if you keep working after the instructor calls for papers.
- You may not use any electronic or computing devices, including calculators, cell phones, smartwatches, etc. You may receive a deduction if we hear or see your electronic device during our exam time.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- This is a closed-note exam, but you may use the provided cheat sheet for reference. As noted on the cheat sheet, you may assume id, qs, qsa, gen, and checkStatus aliases are provided in client-side JS (see cheatsheet for reference)
- Do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...).
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.
- If you are done early, please pack up quietly and turn your exam into a TA as you leave. You will not be allowed to return once you have exited the exam room.

`*('0')* Good luck! *('0')*`

Question	Score	Possible
Short Answers		10
Node.js Web Service		16
JS with AJAX		12
SQL		12
XC		1
Total		50

1. Short Answers (10pts)

1. Callbacks vs. Promises. Most functions in the fs module are asynchronous by default, accepting a callback function as the last argument. We use the `util` module when we want to promisify these functions (e.g. **fs.readFile**):

```
const util = require("util");
const fs = require("fs");
const readFile = util.promisify(fs.readFile);
```

Provide one advantage of using promisified versions of callback-based functions like `fs.readFile`. For full credit, you must clearly identify the advantage between the two options, and may refer to other promisified functions we've covered in your justification.

2. Regex. Circle the strings which match the regex pattern shown below:

`/^[A-Z][a-z]+.[a-z]{3}$/`

- `Helloworld.foo`
- `A.biz`
- `Apple@com`
- `Abc.d3`

Refer to the provided cheatsheet for regex reference if needed.

3. Node/SQL Connection. Which of the following statements are true about **`MySQL`** databases and the **`promise-mysql`** functions? Circle all true statements.

- A database can contain more than one table.
- If an error occurs in **`mysql.createConnection`**, the returned db object will still be defined.
- If an error occurs in **`db.query`**, the db object will still be defined.
- When we have a db object used in an Express endpoint function, we must **always** call `db.end()` **before** using **`res.send()`**
- When we have a db object used in an Express endpoint function, we must **always** call `db.end()` **after** using **`res.send()`**

4. SQL TABLE Relationships. Which of the following statements are true about **`FOREIGN KEY`**s and **`PRIMARY KEY`**s? Circle all true statements.

- Every table must have at least one **`PRIMARY KEY`**.
- A table can have more than one **`FOREIGN KEY`**.
- **`PRIMARY KEY`**s must be **`INT`**s
- **`FOREIGN KEY`**s must reference either a **`PRIMARY KEY`** or a **`UNIQUE`** column in another table.
- A **`PRIMARY KEY`** referenced by a **`FOREIGN KEY`** can have the same column name.
- A **`PRIMARY KEY`** referenced by a **`FOREIGN KEY`** can have a different column name.

5. Asynchronous Program Execution. Circle the option which represents the console output of this program:

```
function init() {
  console.log("foo");
  hello();
  syncFunction();
}

async function hello() {
  await resolveInTwoSeconds();
  console.log("bar");
}

async function resolveInTwoSeconds() {
  // this function will always resolve in two seconds, but has no output.
}

function syncFunction() {
  setTimeout(() => {
    console.log("baz");
  }, 1000);
}

init();
console.log("mumble");
```

a	b	c	d	e
mumble foo baz bar	foo baz bar mumble	foo mumble baz bar	foo bar baz mumble	foo mumble bar baz

2. (Node.js Web Service) Books on a Budget (16pts)

See Reference Book Problem 2/3 section for details.

Finish the **app.js** below for Endpoint 1 and Endpoint 2 documented in the spec booklet. For full credit, your app.js **must always send the endpoint's response**, **may not overwrite content headers**, and **may not modify the response after it has been sent**. You do not need to use all required modules, but they are available as needed. For **`glob`** and **`fs`**, you may choose to use the provided promisified functions or the callback versions in the respective modules.

```
"use strict";
const express = require("express");
const util = require("util");
const glob = require("glob");
const fs = require("fs").promises;
const path = require("path");
const globPromise = util.promisify(glob);

const SERVER_ERROR_MSG = "Something went wrong on the server, please try again later.";

const app = express();
app.use(express.static("public"));

// Part A: Implement GET /courses endpoint
```

```
// Part B on following page.
// Part B: Implement GET /books/:course endpoint below
```

```
const PORT = process.env.PORT || 8000;
app.listen(PORT); // end app.js
```

3. **(Client-side JS with AJAX)** Books on a Budget (12pts)

See Reference Book Problem 2/3 section for details.

Write your solution to Problem 3 below:

```
"use strict";
(function() {


  window.addEventListener("load", init);

  // Write your solution here, starting with init function.









// More room provided on next page.

// Continue your solution to Problem 3 as needed here.
```

```
})();
```

4. (SQL) Who needs .txt-books anyways? (12pts)

See Reference Book Problem 4 section for database details.

a (3pts). Write a SQL SELECT query which will return all courses in the **requirements** table, ignoring duplicate course names and sorting in alphabetical order. Do not make assumptions about the rows in the table, but an example using the same data in the reference tables is provided:

Example result:

course
CSE142
CSE154
HCDE308
PSYCH200

Write your SQL query below:

4B (3pts) and 4C (6pts) are provided on next page.

b (3pts). It seems like our customers don't trust sellers who claim to sell $0.10 books. Write the appropriate SQL statement **(not a SELECT query)** which will delete all records from the **offers** table with a price of 0.10 or less.

Write your SQL statement below:

c. (6pts) With this new database design, we can utilize multi-table queries to reimplement our textbook service. Write **a multi-table SELECT query** that will return the book **name**, **seller**, and **price,** and associated **course** of all books required by a course. Order the results by **course** name alphabetically and breaking ties by offered **price** descending. **Do not hard-code any column values. Hint: You will need to use all three tables in a single query and may use the WHERE syntax or JOIN syntax for the query, as long as it's correct.**

Example result:

name	seller	price	course
Java for Duckies	BOOKS 4 CHEAP	0.05	CSE142
Java for Duckies	University Book Store	199.99	CSE142
A Dog's Guide to Web Development	University Book Store	199.99	CSE154
A Dog's Guide to Web Development	PetCo	200.01	CSE154
A Crash Course in Theoretical JS	Books, Etc.	10.01	CSE154
Where Wizards Stay up Late	Books, Etc.	19.99	CSE154
Design of Everyday Things	AMAZON	10.43	HCDE308
...			

Write your SQL query below:

X. Extra Credit (1pt)

For this question, you can get 1 point of extra credit (demonstrating at least 1 minute's worth of work and being appropriate in content).

If you could feature a product in your Final Project e-commerce store dedicated to your TA, what would it be? Your answer may be in text or a drawing.