1 CS 480W/580W Final

Date: Dec 14, 2018 **Max Points**: 100
Time: 120 minutes.
Open-book, open notes; **no electronic devices**

Important Reminder As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

Background:

1. A student needed to extract the body of a HTTP request. The student observed that the HTTP headers in a sample request contained 271 characters and proceeded to use the following code to extract the body of a `request`:

   ```
   const requestBody = request.slice(271);
   ```

   What is wrong with this approach? How would you fix the problems? (Actual code is not necessary, it is sufficient to clearly describe the problem and the suggested fix). *10-points*

2. A web service is called using **doService**`(url, succ, err)` where:

   url Specifies the URL for the web service.

   succ This is a callback function of one argument which is called with the result of the web service when the service succeeds.

   err This is a callback function of one argument which is called with an error object when the service fails.

   Write a wrapper function **wrappedService**`(url)` which wraps the above **doService**`()` so that the wrapper can be called as follows:

   ```
   try {
     const result = await wrappedService(url);
     //code to handle result
   }
   catch (err) {
     //code to handle web service error
   }
   ```

   Provide the code for **wrappedService**`()`. *15-points*

3. Given the following attempt at a `react.js` component for a Binghamton University **B-Number**:

```
01    class BNumber {
02      constructor(props) {
03        this.state = {
04          input: '',
05          error: '',
06        };
07      }
08
09      onBlurHandler(event) {
10        const input = this.state.input;
11        if (!input.match(/^B\d+/)) { //validate
12          this.state.error = 'invalid B-number'
13        }
14      }
15
16      render() {
17        return (
18          B-Number:
19          <input onBlur={this.onBlurHandler}>
20          <br>
21         <span class="error">{this.state.error}</span>
22        );
23      }
24
25    }
```

You may assume that all necessary libraries have been included.

Identify bugs and inadequacies in the above implementation of `BNumber`. *15-points*

4. During a code review, a colleague flags some lines from your code for a react.js component as constituting a major security flaw:

```
...
validate() {
  ...
  if (!input.match(...)) { //validate user input
    this.setState({error: `bad input ${input}`});
  }
  ...
}

render() {
  return (
    ...
    {this.state.error}
    ...
  );
}
```

   Your colleague claims that because you are permitting unvalidated, unescaped user input to be added via the error message to the HTML page, a cracker can craft an input which allows execution of arbitrary JavaScript on your web page. Explain why your colleague is wrong. *10-points*

5. You are given a JavaScript list `gameScores` of combined scores of NBA games where each element of the list has keys `teams`, `date` and `score` and is of the form:

```
{
  teams: Teams, //string of form WinningTeam-LoosingTeam
  date: Date,   //string of form YYYY-MM-DD
  score: Score, //integer giving sum of both team scores
}
```

Example data:

```
const GAME_SCORES = [
  { teams: 'Pistons-Nuggets',
    date: '1983-12-13',
    score: 370
  },
  { teams: 'Pistons-Lakers',
    date: '1950-11-22',
    score: 37
  },
  { teams: 'Mavericks-Blazers',
    date: '2018-12-04',
    score: 213
  }
];
```

(a) Critique the above data representation.

(b) Write a function `renderScores(gameScores)` which uses `mustache.js` to render `gameScores` into a HTML table, such that:

- The return value of the function should be a string containing the rendered HTML with top-level element `<table>`.
- The table should have a 3-column heading row with the columns labelled **Teams**, **Date** and **Score** respectively.
- The heading row must be followed by data rows, one for each item in `gameScores` with its `Teams`, `Date` and `Score` in the appropriate column.
- The table must have a CSS class of `gameScores`.
- if `Score < 150`, then the data row should have its CSS class set to `low`.
- if `150 <= Score < 250`, then the data row should have its CSS class set to `mid`.

- if `Score >= 250`, then the data row should have its CSS class set to `high`.

For example, given the above example data, the call **renderGameScores**`(GAME_SCORES)` should return:

```
<table class="gameScores">
<tr><th>Teams</th><th>Date</th><th>Score</th></tr>
  <tr class="high">
    <td>Pistons-Nuggets</td>
    <td>1983-12-13</td>
    <td>370</td>
  </tr>
  <tr class="low">
    <td>Pistons-Lakers</td>
    <td>1950-11-22</td>
    <td>37</td>
  </tr>
  <tr class="mid">
    <td>Mavericks-Blazers</td>
    <td>2018-12-04</td>
    <td>213</td>
  </tr>
</table>
```

modulo whitespace.

You may assume that the `mustache` module has been `require`'d and is available using identifier `mustache`. *20-points*

6. Assuming that a HTML page contains zero-or-more tables formatted as per the previous question:

   (a) Write a jQuery selector to return a jQuery collection which contains all the `<td>` elements containing `high`-level scores from all such tables on the page.

   (b) Write a JavaScript function `highScoreAverage()` whic uses jQuery to average all the `high`-level scores on the page. You may assume that `jQuery` is accessible as usual in the shortcut `$` variable.

   Hint: A jQuery collection `$list` of jQuery objects can be converted to a JavaScript array of DOM objects using `$list.toArray()` and the HTML content of a DOM object `dom` can be extracted using the property `dom.innerHTML`. *15-points*

7. Discuss the validity of the following statements. What is more important than whether you ultimately classify the statement as **true** or **false** is your justification for arriving at your conclusion. *15-points*

   (a) Mustache will always escape any HTML characters in rendered strings.

   (b) The cache time specified for images should be the same as that for their containing HTML page.

   (c) It is ok for a client to attempt to `DELETE` the same resource multiple times.

   (d) If the propagation of a DOM event is stopped in its bubble phase, the event will never enter its capture phase.

   (e) If a promise is rejected before a `catch` handler is attached to the promise, then the `catch` handler will not be run.