CS312 Spring 2019 – Midterm Solution

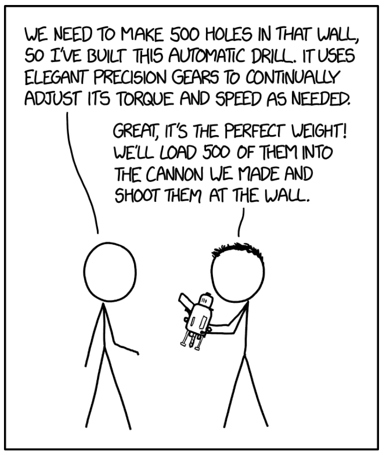
Name: ________________________________

Date: ____________________ **Start time:** ________________ **End time:** ________________

Honor Code:

Signature: ______________________________

This exam is closed book, closed notes, closed computer, etc. You may only use a single double-sided letter-sized sheet of notes. **You have 2 hours in a single sitting.** Read the problem descriptions carefully and write your answers clearly and *legibly* in the space provided. Circle or otherwise indicate your answer if it might not be easily identified. You may use extra sheets of paper, stapled to your exam, if you need more room, as long as the problem number is clearly labeled and your name is on the paper. If you attached extra sheets indicate on your main exam paper to look for the extra sheets for that problem.


WE NEED TO MAKE 500 HOLES IN THAT WALL, SO I'VE BUILT THIS AUTOMATIC DRILL. IT USES ELEGANT PRECISION GEARS TO CONTINUALLY ADJUST ITS TORQUE AND SPEED AS NEEDED.
GREAT, IT'S THE PERFECT WEIGHT! WE'LL LOAD 500 OF THEM INTO THE CANNON WE MADE AND SHOOT THEM AT THE WALL.

HOW SOFTWARE DEVELOPMENT WORKS

Question	Points	Score
JavaScript	11	
React	18	
Agile Practices	19	
Client Server	14	
Data Modeling and Databases	12	
Total:	74	

Question 1: JavaScript [11 points]

(a) Describe the type and length of the return value for each of these methods invoked on an `Array` of length `n`, e.g. `slice` returns an `Array` of length $\leq$ `n`.

i. `map` __`Array` **of length** `n`__

ii. `reduce` __**A scalar value**__

iii. `filter` __`Array` **of length** $\leq$ `n`__

iv. `forEach` __`undefined`__

(b) Choose ONE answer. Helen is creating a JavaScript (JS) application to keep track of her favorite books and plays. She implemented separates JS classes for `Book` and `Play`, because even though both have a `title` and `author`, they are otherwise very different: for example, a `Play` has a `numberOfActs` and other properties that don't apply to a `Book`. Helen needs to sort a combined array of books and plays by their title. What design should she use to solve this problem in JS?

◯ `Book` and `Play` must be subclasses a common class, e.g. `Manuscript` that has the `title` property.

[✓] **As long as `Book` and `Play` both have an attribute named `title`, Helen can use the existing `Array.sort` method on the combined array.**

◯ Helen must create a custom sort function that can handle the combined array by checking the type of each item as part of the comparison function.

◯ Helen must separate the collection into two collections (one containing only Books and the other only Plays), sort each collection, then merge the two sorted collections.

(c) Choose ONE answer. Which of the following Jest features helps us implement tests that satisfy the "I" in F.I.R.S.T.?

◯ The "watcher" that automatically runs the tests when a file changes

◯ The many different matchers, e.g. `toBeFalsy`.

◯ Multiple `expect` calls in a single test

[✓] **The `beforeEach` and `afterEach` functions**

(d) Choose ONE answer. Assume `articlesToSections` returns a sorted array of Simplepedia section labels for an array of articles. The following test

```
expect(articlesToSections(samplesArticles)).toEqual(['A','B']);
```

is an example which of the following kinds of testing?

[✓] **Unit testing**

◯ Integration testing

◯ End-to-end testing

(e) `waitSeconds(sec)` returns a promise that resolves after `sec` seconds have elapsed. Write example valid output from running this code with `node`. Make sure to include the correct label, e.g. “Delay 1”, with each statement.

```
waitSeconds = (sec) => {
  return new Promise(resolve => setTimeout(resolve, sec * 1000));
};
elapsedSeconds = (start) => Math.floor((Date.now() - start) / 1000);

let current = Date.now();
waitSeconds(2)
  .then(() => {
    console.log(`Delay 1: ${elapsedSeconds(current)}s`);
    return waitSeconds(3);
  })
  .then(() => {
    console.log(`Delay 2: ${elapsedSeconds(current)}s`);
    return 4;
  })
  .then(() => {
    console.log(`Delay 3: ${elapsedSeconds(current)}s`);
  });
console.log(`Delay 4: ${elapsedSeconds(current)}s`);
```

Solution:

```
Delay 4: 0 s
Delay 1: 2 s
Delay 2: 5 s
Delay 3: 5 s
```

Question 2: React [18 points]

(a) There are several correctness problems with this React to-do list component (i.e. not just poor style). Identify three (3) such problems. Each problem should be distinct, not just a variation on the same problem. You do *not* need to provide a fix for the problems you identify.

```
import React, { Component } from 'react';
class ToDo extends Component {
  constructor(props) {
    super(props);
    this.state = { truncate: false, newItem: '' };
  }
  render() {
    if (this.props.items.length > this.props.numToShow) {
      this.setState({ truncate: true });
    }
    const showItems = this.state.truncate ?
      this.props.items.slice(0, this.props.numToShow) :
      this.props.items;
    return (<div>
      <ul>{showItems.map(item => <li key={item}>{item}</li>)}</ul>
      <p onClick={() => {
        this.setState(prevState => ({ truncate: !prevState.truncate }));
      }}>...</p>
      <input value={this.state.newItem} onChange={(evt) => {
        this.state.newItem = evt.target.value;
      }}/>
      <button onClick={() => {
        this.props.items.push(this.state.newItem);
      }}>Add</button>
    </div>);
  }
}
```

Solution:

1. Setting state in the render method has the potential to create an infinite loop as `setState` will trigger a re-render.
2. The `onChange` callback mutates state without invoking `setState` to notify React to trigger re-rendering.
3. The `onClick` handler mutates the props instead of accepting and invoking a callback from the parent component.

(b) The `ToDo` component is designed for quick input, with a text input and an Add button at the bottom of the component. To support data persistence, the application sends new to-do items to a server via the `fetch` function. Imagine that a teammate has already implemented a suitable mock for `fetch`. Ignore the current (broken) implementation, and think about performing integration testing based on how the user would experience adding a new item.

i. Using pseudo-code or detailed text, explicitly describe each step of a test of adding a new item to the to-do list.

Solution: For full credit, the solution should have four parts: mounting of the component, a test of the precondition, a simulation of the user interaction, and test of the post-condition (depending on how `fetch` has been mocked, there may need to be a wait, or a flushing of refreshes, but we will ignore that implementation detail). Such a test would contain the following steps:

1. Mount the component (or whole application)
2. Check that the new item is not already in the list
3. Simulate adding the new item text to the input box
4. Simulate clicking the Add button
5. (Optional) Test that the `fetch` mock was invoked with the correct arguments
6. Check that the new item is now in the list

ii. The `ToDo` component has a feature that restricts the number of visible elements. Propose two (2) additional tests for your test suite that cover this behavior. You do *not* need to provide implementation details of these tests, just summarize the behavior that needs to be tested.

Solution:

- An `this.props.items` array with more than the specified number of values to show will result in items being hidden
- Clicking on the ellipse will show the entire list (and truncate it again).

(c) As part of your to-do list application you are developing a separate `DatePicker` component that provides a calendar view to help your users pick "due dates" for their to-dos. `DatePicker` will be rendered next to the text input field. Briefly describe what state you will need and where that state will be maintained.

> **Solution:** We will need to maintain the date as state, and that state should be maintained in a parent component, e.g. the `ToDo` component. The date picker should communicate any changes, i.e. the date the user picks, back up to the parent component via a callback.

Question 3: Agile Practices [19 points]

(a) The requirements specification in Plan and Document corresponds to what Agile practice(s)?

> **Solution:** User stories and associated scenarios.

(b) Choose ONE answer. You have an idea for a new feature for your project. According to our CS312 development process, what is your next step?

- ◯ Implement a spike to see if it will work
- ◯ Add it to the project backlog
- ◯ Draw a "lo-fi" storyboard
- [✓ **Write a user story**]
- ◯ Pitch it at the next stand-up

(c) In the Scrum approach, fill in who has responsibility for each of the following tasks:

i. Prioritizing features **Product Owner**

ii. Facilitating the sprint planning meeting **Scrum Master**

iii. Updating the sprint backlog during the sprint **Development team**

iv. Determining who implements which backlog items **Development team**

(d) Briefly describe three (3) motivations for creating a pull request (PR) as part of our project workflow.

Solution:

1. Your teammates have an opportunity to see the new code so they can start thinking about how that code might affect their own work.
2. PRs provide a mechanism for code review.
3. Travis-CI automatically tests the PR to ensure the resulting merge will not "break" the application (before you merge the new code into `master`).

(e) Colors in CSS are typically specified in hex, e.g. `#0D395F` for Panther blue.

i. Write a user story in Connextra format for a new feature for the color picker that displays the current color in hex.

Solution: As a user,
I want the color displayed in hex (e.g. `#0D395F`) that could be copied directly into CSS,
so that I can be more efficient when developing CSS.

ii. Write a Gherkin-style scenario for this feature that incorporates user interaction.

Solution: Given the color components are all 0,
When the user changes the sliders to `{ red: 13, green: 57, blue: 95 }`)
Then the hex display text should be `#0D395F`.

Question 4: Client Server [14 points]

(a) You are developing a web application with a contact information form. One of your teammates notices that the client-side code validates the inputs (makes sure fields are present, checks that email addresses and phone numbers are in valid formats) before sending it to the server, which then performs essentially the same validations. Your teammate suggests removing either the server- or client-side validation steps to DRY out the codebase and make it more maintainable. Make the counter argument that advocates leaving both server- and client-side validation in place by providing advantages for each step that can't be duplicated by the other.

> **Solution:** Validating on the client can provide a better user experience and simplify control flow. The user gets immediate feedback (close in time) on potential problems making it easier for them to fix any mistakes.
>
> We must validate on the server because we can never trust data from the client. JavaScript running client-side can be manipulated or "spoofed" with another client (e.g. `curl`, so the server can never be sure if it is getting data from the original client code executing as intended. Accepting unknown data without validation makes our applications insecure.

(b) Select ALL that apply. Given the HTTP request `GET http://www.example.com:8000/search?q=cs312`, which elements of that request does Express use to determine which handler to execute?

- [✓ **GET**]
- ◯ www.example.com
- ◯ :8000
- [✓ **/search**]
- ◯ q=cs312

(c) For each of the following actions in a web application provide a appropriate RESTful HTTP verb and URL, e.g. `GET /api/articles/3`.

i. Read the data for a single to-do list item

> **Solution:** `GET /todos/3`

ii. Create a new movie

> **Solution:** `POST /movies`

iii. Update a single to-do list item

> **Solution:** `PUT /todos/3`

iv. List all of a user's movie ratings

> **Solution:** `GET /users/3/ratings`

Question 5: Data Modeling and Databases [12 points]

(a) Select ALL that apply. Which of the following statements are true regarding the use of migrations to manage the database schema of a SaaS app?

- √ **Migrations allow "versioning" the database schema analogously to how code is versioned.**
- √ **Migrations can be expressed in a way that is independent of minor syntactic differences between different underlying databases.**
- ◯ Migrations may change the schema, but they never result in destroying or discarding data.

(b) You are writing a simple blogging application in a which users can write posts and comment on posts that they or others wrote. As with Simplepedia, a post should have a `title`, `body` and `edited` time. A comment should have a `body` and `edited` time.

i. Draw the CRC cards for the nouns in this application focusing just on posting and commenting.

Solution:
The intended answer included the Post, Comment and User nouns (models), but not login/logout/etc. responsibilities (hence the "focusing just on posting and commenting"). Because that "focusing" was ambiguous, full credit was awarded for just the Post and Comment nouns.

User	
Knows posts	Post
Knows comments	Comment
Knows commented posts	Comment, Post
Create post	Post

Post	
Knows author	User
Knows comments	Comment
Knows title, body and edited	
Add comment	User, Comment

Comment	
Knows author	User
Knows post	Post
Knows body and edited	

ii. What are the association(s), if any, between the various nouns?

Solution:

- A `User` has many `Posts` and a `Post` belongs to a `User`, i.e. a "one-to-many" relationship
- A `Post` has many `Comments` and a `Comment` belongs to a `Post`, i.e. a "one-to-many" relationship
- A `User` has many `Comments` and a `Comment` belongs to a `User`, i.e. a "one-to-many" relationship
- A `User` has many commented `Posts` through `Comments`, i.e. a "many-to-many" relationship

Page purposely left blank.