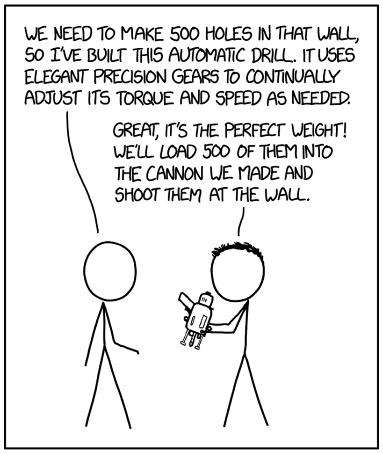
CS312 Spring 2020 – Midterm Solution

Name: ________________________________

Honor Code:

Signature: ______________________________

This exam is open notes, open course notes and open slides but no other sources are permitted (e.g. Google, StackOverflow, and other resources, even if linked from the course page, are not permitted). Attempting to execute any of the code samples is not permitted. There is no time limit, **but you must submit your completed exam to Gradescope by the deadline** (think of this as like a homework assignment that you complete entirely on your own). You can print and complete the exam using this template, or on separate sheets of paper. Read the problem descriptions carefully and write your answers clearly and legibly. Circle or otherwise indicate your answer if it might not be easily identified.


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	24	
React	24	
Agile Practices	17	
Client Server	15	
Data Modeling and Databases	14	
Total:	94	

Question 1: JavaScript [24 points]

(a) (6 points) 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 ≤ `n`.

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

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

iii. `filter` `Array` **of length** ≤ `n`

iv. `forEach` `undefined`

(b) (2 points) Choose ONE answer. Helen is creating a JavaScript (JS) application to keep track of her favorite books and plays. She implemented separate 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?

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

B. As long as `Book` and `Play` both have an attribute named `title`, Helen can use the `Array.sort` method with a callback that compares titles.

C. Helen must create a custom sort function that invokes different comparison functions depending on the types of the values to be compared.

D. 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) (2 points) Choose ONE answer. Which of the following Jest features helps us implement tests that satisfy the "R" in F.I.R.S.T.?

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

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

C. Multiple `expect` calls in a single test

D. The `beforeEach` and `afterEach` functions

(d) (6 points) Assume we wrote a helper function for Simplepedia, `toSection`, that takes an article as its argument and returns the capitalized first letter of the title (e.g. its section). Write a single F.I.R.S.T. unit test using Jest for that function.

Solution:

```
expect(toSection({ title: "a title" })).toEqual('A');
```

(e) (8 points) `waitSeconds(sec)` returns a promise that resolves after `sec` seconds have elapsed. `elapsedSeconds(date)` returns the time in seconds between now and the `date` argument. Write example valid output from running this code with `node`. Make sure to include the correct label, e.g. “Delay 1”, with each statement.

```
let current = Date.now();
console.log(`Delay 1: ${elapsedSeconds(current)}s`);
waitSeconds(3)
  .then(() => {
    console.log(`Delay 2: ${elapsedSeconds(current)}s`);
    return waitSeconds(2);
  })
  .then(() => {
    console.log(`Delay 3: ${elapsedSeconds(current)}s`);
  })
  .catch(() => {
    console.log(`Delay 4: ${elapsedSeconds(current)}s`);
  });
console.log(`Delay 5: ${elapsedSeconds(current)}s`);
```

Solution:

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

Question 2: React [24 points]

(a) (8 points) There are several correctness problems with this React component for displaying and creating to-do list items (i.e. not just poor style). Identify three (3) such problems referencing line numbers as appropriate. 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.

```
 1  import React, { useState } from 'react';
 2  function ToDo(props) {
 3    const [newItem, setNewItem] = useState('');
 4
 5    const truncate = true;
 6    const showItems = truncate ?
 7      props.items.slice(0, props.numToShow) :
 8      props.items;
 9
10    return (<div>
11      <ul>{showItems.map(item => <li key={item}>{item}</li>)}</ul>
12      <p onClick={() => { setTruncate(!truncate); }}>...</p>
13      <input value={newItem} onChange={(evt) => {
14        newItem = evt.target.value;
15      }}/>
16      <button onClick={() => { props.items.push(newItem); }}>Add</button>
17    </div>);
18  }
```

Solution:

1. The `onChange` callback on line 14 mutates state without invoking `setNewItem` to notify React to trigger re-rendering.
2. The `onClick` handler on line 16 mutates the props instead of accepting and invoking a callback from the parent component.
3. `truncate` is used like state, but not created with a `useState` hook, thus `setTruncate` (on line 12) is not defined.

(b) (8 points) The `ToDo` component is intended to have a feature that restricts the number of visible elements if the `items` prop is longer than some threshold. The user can click the ellipsis (`...`) to toggle the truncation, i.e. show all the items or just a subset. Write a Gherkin-style test scenario(s) that covers this toggling behavior. You do *not* need to provide the implementation details of the tests, just describe the scenario for the test.

Solution:
Clicking on the ellipsis will show the entire list (and truncate it again):

```
Given that there are 10 to-do items, numToShow is 5 and truncate is true
Then only the first 5 to-dos should be shown
When I click on the elipsis
Then all 10 to-dos should be shown
When I click on the elipsis
Then only the first 5 to-dos should be shown
```

(c) (8 points) As part of your to-do list application, you want to automatically convert any correctly formatted links in the to-do text into an actual "clickable" link, i.e. an HTML '`<a href="..."></a>` element. Briefly describe how would implement that feature in your React application (no code is expected, just a description of your approach). You may propose other React components if appropriate. Your solution will be evaluated based on maintainability and testability.

Solution:
No state is required, so we can implement a purely functional component, say named `LinkDisplay`, which will be rendered inside the `<li>` inside the list, e.g.

```
<li key={item}><LinkDisplay item={item} /></li>
```

It will take a single prop, the to-do string, and scan that string for links (using a regular expression or other approach), replacing any links in the string with a `<a href="...">{link_text}</a>` component. No other props are needed. This approach will be more testable and maintainable than directly integrating that functionality into the `ToDo` component because this feature can be tested in isolation and extended with additional automatic formatting without needing to change `ToDo`.

Question 3: Agile Practices [17 points]

(a) (4 points) Proponents of the Scrum process assert that Scrum is at its best when it is difficult to plan ahead. Briefly describe the arguments for that assertion.

Solution:
Teams practicing Scrum, with its short development cycles or sprints (on the order of weeks) that incorporate a planning phase, can quickly respond to new information or changing requirements in a way that the plan-and-document process cannot.

(b) (2 points) 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?

A. Implement a spike to see if it will work
B. Add it to the project backlog
C. Draw a "lo-fi" storyboard
D. Write a user story
E. Pitch it at the next stand-up

(c) (6 points) 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 stay up-to-date with changes to the code so they can start thinking about how that code might affect their own work (i.e. get notifications, can monitor PR list).
2. PRs provide a mechanism for code review where a teammate reviews the code and "signs-off" before it is integrated into the `master` branch.
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`).

(d) (5 points) Write a S.M.A.R.T. user story for a new feature for the to-do list that enables user to mark a to-do as important, changing its font to bold. Your user story will be evaluated on format and quality.

> **Solution:**
> As a user,
> I want to mark some to-dos as important,
> so that they can be easily distinguished from less important to-dos by their bold type.

Question 4: Client Server [15 points]

(a) (5 points) In Simplepedia we purposely did not make any changes to the local article collection until we received a response from the server. Briefly describe why we chose this approach, and did not, for instance, immediately update the collection state after launching the `fetch` request.

> **Solution:**
> The client does not have sufficient information to assign the articles an `id` or ensure invariants like all articles have unique titles. Only the server can enforce those requirements. By not modifying our state until a request has successfully completed and we get the data back from the server, we ensure we don't update our local collection with invalid or incomplete (or out-of-date) data.

(b) (2 points) Select ALL that apply. Given the HTTP request
`GET http://www.example.com:8000/assignments?type=practical`,
which elements of that request does Express use to determine which handler to execute?

- **A. GET**
- B. www.example.com
- C. :8000
- **D. /assignments**
- E. type=practical

(c) (8 points) 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 to-do items

Solution: `POST /todos`

iii. Remove a single to-do list item

Solution: `DELETE /todos/3`

iv. List all of a user's to-do items

Solution: `GET /users/3/todos`

Question 5: Data Modeling and Databases [14 points]

(a) (4 points) Identify the models you would define in your server backend to implement the following user story:

As a user, I want to create a to-do item with a due date, so that I can remember to complete all of my tasks on time

Solution:
The two models would be `User` and `ToDo`. Due date would be implemented as an attribute of `ToDo`, not as a separate model.

(b) (2 points) Choose ONE answer. You want to extend Simplepedia to track the user who authored an article. To do so you define a `User` model. Which association between `User` and `Article` best models this relationship?

A. A `User` has one `Article`, an `Article` belongs to a `User`.

B. A `User` has many `Articles`, an `Article` belongs to a `User`.

C. A `User` has many `Articles`, an `Article` belongs to many `Users`.

D. A `User` has many `Articles` through `Authorship`, an `Article` has many `Users` through `Authorship`.

(c) (8 points) In addition to authors, you also want to track which users have edited an article and when they did so (only the user and time, you don't need to track how they edited the article). What association between `User` and `Article` would model this relationship. Your answer should be formatted similarly to the answers to the previous part. In your answer note which Model would "know" the time that each user edited the article.

Solution:
There is a many-to-many relationship between the `User` model and the `Article` model, i.e. a user edits many articles and an article is edited by many users. To store the edited time we would introduce a new model `Edited`, similar to the `Ratings` model in the Film Explorer, i.e. `Edited` "knows" the edited time. The association would be a A `User` has many `Article`s through `Edited`, an `Article` has many `User`s through `Edited`.