CS312 Spring 2025 – Midterm

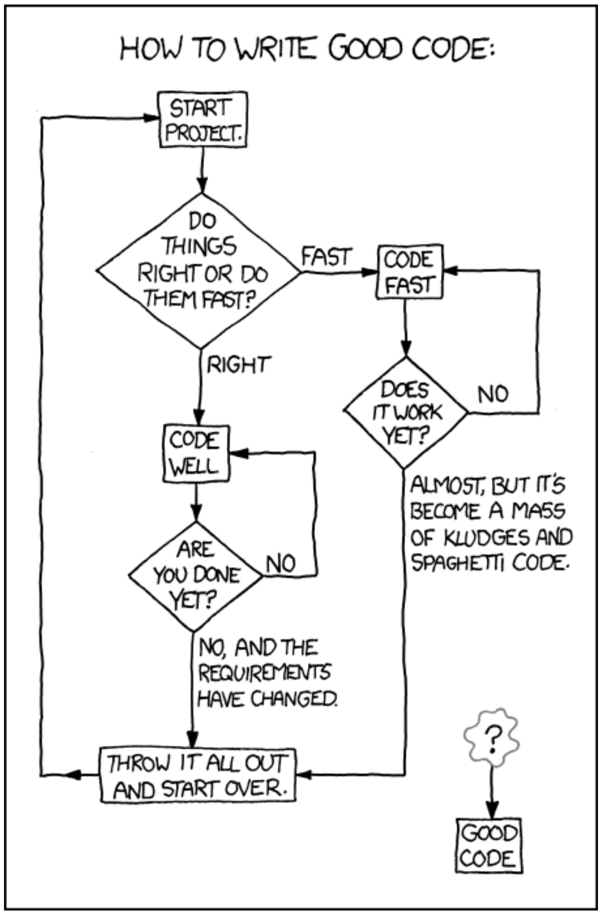
Name: ______________________________

Date: _______________ **Start time:** ____________ **End time:** ____________

Honor Code:

Signature: ______________________________

This exam is closed computer, course web page, notes, texts, searching online, AI and consulting with others, i.e., closed everything except a notes page you prepare. You are only permitted *one* double-sided letter-sized sheet of notes of your own creation (may be typed). **You have 2 hours in a single sitting to complete the exam.** 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.


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
?
THROW IT ALL OUT AND START OVER.
GOOD CODE

Learning Target	Assessment
1	
2	
3	
4	
5	
6	
7	
8	

Question 1. User stories

You are developing a web application for organizing public events. When interviewing stakeholders, multiple respondents described wanting a calendar view that summarizes the number of events on each day of date range. Write two I.N.V.E.S.T. user stories for this feature, one from the perspective of an event planner, the other from the perspective of a potential event goer. Your user stories will be evaluated on format and quality.

(a) Event planner:

(b) Potential event goer:

Question 2. Javascript

Assume the function `wait(sec)` returns a promise that resolves after `sec` seconds have elapsed. Assume that every other operation is instantaneous (e.g., takes 0 milliseconds). Remember that `Date.now()` returns the number milliseconds elapsed since 12:00AM January 1, 1970, UTC.

```
function do(time, since) {
  return wait(time).then(() => {
    console.log(Date.now() - since);
  });
}

const start = Date.now();
do(2, start);
do(1, start).then(() => {
  do(4, start);
});
do(3, start);
console.log(”end”);
```

```
async function do(time, since) {
  await wait(time);
  console.log(Date.now() - since);
}

const start = Date.now();
do(2, start);
do(1, start);
do(4, start);
await do(3, start);
console.log(”end”);
```

Consider the two code snippets above. Write the expected output for left-side code below on the left. If the right-side code produces the same result indicate below, otherwise provide the expected output below on the right.

◯ Both snippets produce the same output

Question 3. Testing

You are developing a React component named `WordGame` for playing a word guessing game. The component takes the hidden word as a prop. The component provides a text input where the user can enter their guess and button "Guess" to check their guess. Incorrect guesses are displayed in a list below the input with an × appended. A correct guess is displayed at the end of the list with a ✓ appended. When the user guesses correctly the component displays the time elapsed since the game started (the component was mounted), e.g.,"You guessed in 5.2s!". Using the skeleton below, implement **pseudo-code** for a F.I.R.S.T. unit test to verify that game correctly handles a correct guess after one incorrect guess. You do **not** need to provide executable Javascript, instead describe the steps of your test as pseudo-code. For example, one of the steps in your pseudo-code might be:

Assert mock function was not called

You may or may not need all of the functions below. You only need to include pseudo-code in bodies of the functions relevant to your answer.

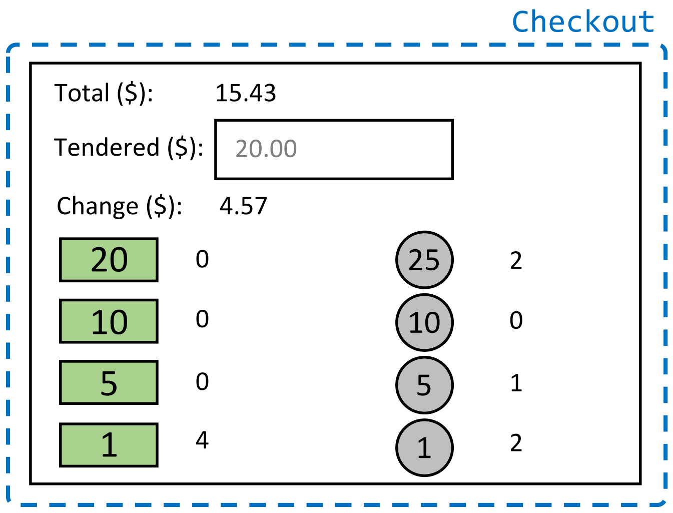
```
describe("Word game", () => {
  beforeEach(() => {









  });
  afterEach(() => {









  });
  test("Shows correct end of game after one incorrect guess", () => {

















  });
});
```

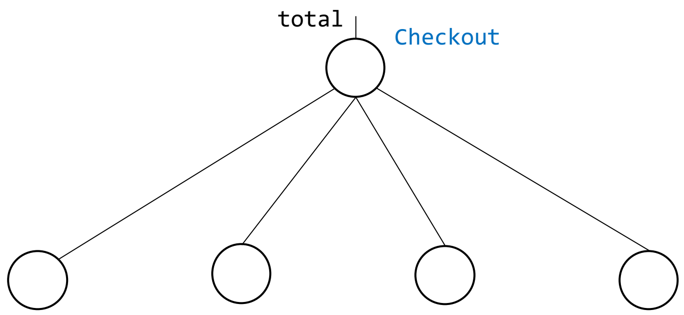
Question 4. Scenarios

On your application's login page the user can toggle the password visibility, i.e. switch between masked characters and plain text, by clicking on an icon in the password field. When the password is hidden, a "Show password" icon appears, when it is in plain text a "Hide password" icon appears. Write a Gherkin-style test scenario for changing the password visibility. You do not need to provide the implementation details of the tests, just describe the scenario for the test.

Question 5. React

You are implementing the `Checkout` component shown below with React. It receives a numeric prop representing the total purchase. Entering an amount paid (tendered) by the customer in cash computes the necessary amount of change and how the change should be provided, i.e., the number of bills and coins of the denominations shown. Outline and label the wireframe (below, top) with a possible set of components. Label the tree (below, bottom) with components to show the hierarchy. Label the tree nodes with state implemented in that component and label the tree edges with props passed to each component (similar to the figure in programming assignment 2). Repeated components can be labeled once in the tree. The top-level component and its prop(s) are labeled for you. Any implementation reflecting good React practices will be accepted. You may not need all the nodes in the tree or may need to add nodes depending on your design; cross out any unused nodes. Your component, state and prop names should be sufficiently descriptive that their role is clear.





Question 6. REST

For each of the following pages in a NextJS-based event management application, provide an appropriate RESTful front-end (browser) URL for that page and, where relevant, an appropriate RESTful server API endpoint (HTTP verb and URL) that component would interact with. An example is provided below.

Page	Page URL	API HTTP verb and URL
View all articles on Simplepedia	**/articles**	**GET /api/articles**
Search for events near a zipcode	____________________	____________________
Update an event details	____________________	____________________
Add a comment to an event	____________________	____________________

Question 7. Data modeling

You are developing a web application for organizing public events through which people can find and register to attend events. You will be using a relational database to store the data for this application.

(a) Identify the *minimum* set of models you would define in your server backend to implement the following user story:

As a past attendee, I want to review events that I have registered for and attended, so that I can provide feedback to the event organizers and information to future potential attendees.

(b) Which of the following best describe the relations between the following pairs of entities. Select one answer for each pair, then briefly explain your answers.

User and Event	Event and Review
○ One-to-One	○ One-to-One
○ One-to-Many	○ One-to-Many
○ Many-to-Many	○ Many-to-Many
○ No relation	○ No relation

(c) In a normalized schema designed for a relational database (RDBMS), what schema would be needed to implement the user story in part (a)? You do not need to provide SQL, just the attributes, their types, the primary key, and any foreign key constraints needed to implement the user story. You only need to provide the schema relevant to the user story, not the entire schema for the application.

Question 8. Development processes

For each of the following, indicate whether the action would be consistent with the best practices for software development as described in class or not consistent. Here "consistent" is defined as consistent with good development practices generally, not that it was required as part of our class. Briefly explain each answer.

(a) Keep a technically complex feature as a single user story to ensure continuity during its development, especially across sprints.

◯ Consistent ◯ Not consistent

(b) Regularly rebase the `main` branch to simplify and linearize the project's commit history.

◯ Consistent ◯ Not consistent

(c) Configure your Continuous Integration (CI) test pipeline to treat any warnings as errors, i.e., the CI pipeline fails if any warnings are generated.

◯ Consistent ◯ Not consistent