CS 343: Application Development
Practice Exam 2

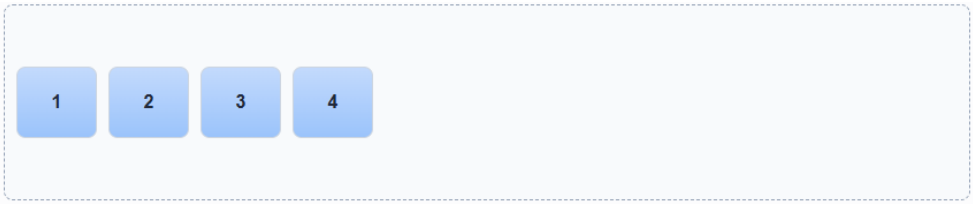
James Madison University
Spring 2026

Name: _____ JACard #: _____

Question 1. CSS Layout
(20 pts)

Given each image, complete the CSS to implement the **layout** using flexbox and grid properties. Assume each item has a fixed size and the parent container has the container class. You do not need to do any additional styling (color, borders, padding/margin, etc.).

a. (5 pts)

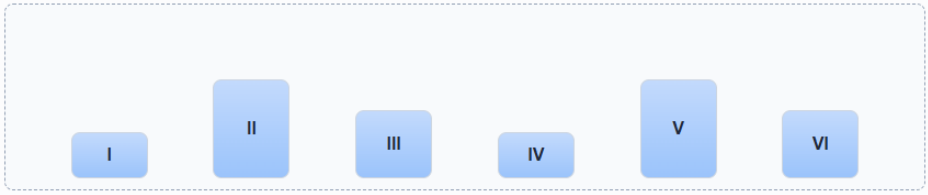



```
.container {
    display: flex;
    flex-direction: row;




}
```

b. (5 pts)




```
.container {
    display: flex;
    flex-direction: row;




}
```

c. (10 pts)

Given the image of the desired layout and the HTML, write the CSS rules to implement the **layout** using flexbox and grid properties. You do not need to do any additional styling (color, borders, padding/margin, etc.).


```
<div class="container">
  <div id="a">A</div>
  <div id="b">B</div>
  <div id="c">C</div>
  <div id="d">D</div>
</div>
```

```
.container {
    display: grid;









}
// Add more CSS rules below as needed
```

Question 2. Array Methods
(20 pts)

For the following problems, you may only use combinations of *.map()*, *.filter()*, *.reduce()* to solve the problem. You **may not** use any loops (*for*, *while*, or even *.forEach()*).

Full credit will only be given for using the appropriate array method(s) for each problem *and* using them appropriately.

a. Write a function *onlyEven* that takes an array of numbers and returns a new array containing only even values.

```
function onlyEven(nums) {




}
```

b. Write a function *addOne* that takes an array of numbers and returns a new array where each number is increased by 1.

```
function addOne(nums) {




}
```

c. Write a function *nameLengths* that takes an array of strings and returns the **sum** of all their lengths.

```
function nameLengths(names) {




}
```

d. Given an array of objects representing vehicles, like the following:

```
[
    { make: "Toyota", model: "Camry", year: 2025 },
    { make: "Honda", model: "Civic", year: 2018 },
    ...
]
```

Write a function *onlyNew* that takes an array of such objects and returns a new array containing only the **models** of vehicles that were made in 2019 or later.

```
function onlyNew(vehicles) {




}
```

Question 3. Fetch and External APIs
(30 pts)

Assume you have an HTML list of track IDs (representing different songs from Spotify):

```
<body>
    ...
    <ul class="trackList">
        <li>3n3Ppam7vgaVa1iaRUc9Lp</li>
        <li>7qiZfU4dY1lWllzX7mPBI</li>
        <li>0VjIjW4GlUZAMYd2vXMi3b</li>
        ...
    </ul>
    ....
</body>
```

Given the API specification in Attachment 1, implement an asynchronous function, **`saveSongs()`**, which must get all the track IDs and make an API request to save tracks for the current user.

Question 4. Local Storage and Query Parameters

3.1 (15 pts) Implement the **`get_data()`** function in `stocks.js`.

3.2 (15 pts) Implement the **`onPageLoad()`** function in `stocks.js`.

Attachment for Question 3

Spotify API Documentation

Base URL: `https://api.spotify.com/v1/`

Save Tracks for Current User

Description: Save one or more tracks to the current user's 'Your Music' library.

Method: PUT

Endpoint: `/me/tracks`

Body: A JSON object containing the following properties:

- **ids** -- array of strings

  A JSON array of the Spotify IDs.

  For example: `["4iV5W9uYEdYUVa79", "eyT98MSxVHPZCA6M"]`

Attachment for Question 4

An application for tracking stock prices uses a third-party API. The data includes company names (like "Apple Inc"), stock symbols (like "AAPL"), and other information.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Stock Info</title>
</head>
<body>
    <div>
        <div id="symbol"></div>
        <div id="shares"></div>
        <div id="price"></div>
    </div>
    <script src="stocks.js"></script>
</body>
</html>
```

A file named `stocks.js` consists of two functions:

1. An *asynchronous* function named **`get_data()`** that takes *one argument* named `key`. The function attempts to load JSON data from local storage using `key`.

- If found, the data (stored as a JSON string) is parsed into an object.
- If not found, the data is fetched using a global variable `API_URL` (assume this exists). The response (in JSON format) is parsed into an object. The object is then saved (as a JSON string) in local storage using `key`.

Either way, the resulting object (loaded from local storage or fetched from the server) is returned.

Hint: Remember to `await` the two asynchronous operations. You will need to use `JSON.parse()`, `JSON.stringify()`, `localStorage.getItem()`, and `localStorage.setItem()`.

2. A function **`onPageLoad()`** that reads the current location's query string, for example:

`?symbol=AAPL&shares=50&price=280.5`

with the keys "symbol", "shares", and "price". The actual values may, of course, differ.

The function must parse the query string and then display those values in the corresponding HTML elements.