Short Answer:

Answer the following questions with complete sentences in **your own words**. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers.

1. What is Node.js?
2. What are some differences between Node.js and JavaScript?
3. What is npm?
4. What is CommonJS? How does it differ from ES Modules?
    1. How do we import files into other files in Node.js?
5. How can you make the server automatically restart when files are modified?
6. What are some global objects in Node.js?
7. Explain how the Node.js architecture is event-driven in terms of event emitters and other core modules.
8. What are streams? In Node.js, what are the different kinds of streams?
9. What is the difference between fs module's readFile and createReadStream?
10. What are the different timing or scheduling functions in Node.js?
11. How does the event loop work in Node.js?
12. How do you manage multiple node versions?
13. Explain the request & response cycle.
14. Explain serialization vs deserialization.
15. What is Express.js?
16. How do you build a basic server in Node.js using express?
17. What is middleware?
18. What are ports? Which one is reserved for HTTP and HTTPS?
19. What is the difference between response.send() and response.write() in express?
20. Explain query strings vs route parameters.

Coding Questions:

Use HTML/CSS/JS to solve the following problems. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. **To-Do App (Express.js)**
   Implement an express server that handles requests for a basic to-do app and interacts with the **file system**. Organize your to-do routes in **a separate module using express.Router**. Be sure to test your routes with Postman and submit screenshots of the server responses.
   - Feel free to implement the front-end (HTML/CSS/JS) if you have time, but it is **not required**.
   - The to-do app should include the following route paths & feature requirements:

   1. `POST: /todo/{filename}`
      The incoming POST request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
      - The request body should contain JSON data that corresponds to a to-do item: **title**, **description**, **status** (In Progress, Completed), and **priority** (Low, Medium, High). The server should generate an additional property, **"timestamp"**, using the current

timestamp.
- If the specified file exists, add the to-do item to the specified json file. Then respond with the new json file contents and appropriate status code.
- If no file exists, create a new .json file with this name and the POST data.
- If there is an error, respond with the appropriate error, message, and status code.

2. `GET: /todo/{filename}`
   The incoming GET request url contains the name of a local .json file that contains all the to-do items. It is up to you if you want to use query strings or route parameters.
   - If the specified file exists, respond with the file contents.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

3. `PUT: /todo/{filename}`
   The incoming PUT request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
   - The request body should contain JSON data with the to-do item's title and new title, description, status or priority.
   - If the specified file exists, modify the corresponding to-do item based on the request body. Respond with the new json file contents and appropriate status code.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

4. `DELETE: /todo/{filename}`
   The incoming DELETE request url contains the name of a local .json file that contains all the to-do items. (**hint: route parameters**)
   - The request body should contain JSON data with the to-do item's title.
   - If the specified file exists, remove that to-do item. Respond with the new json file contents and appropriate status code.
   - If no file exists or there is an error, respond with the appropriate error, message, and status code.

Paired Programming:

Use JS to solve the following problems with your partner. Please remember to label who was acting as the driver and navigator for each problem and write down your feedback on their performance. **Feedback will be confidential**.

Leetcode #205: Isomorphic Strings
Leetcode #206: Reverse Linked List