**Assessment Questions:**

**Question 1: DSA** Given an array of integers, return the length of the longest increasing subsequence. A subsequence is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements. For example, given the array [10, 9, 2, 5, 3, 7, 101, 18], the longest increasing subsequence is [2, 3, 7, 101], and its length is 4.

**OR**

**Question 2: DSA** Given an array of integers nums and an integer target, return the indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

For example, given:

const nums = [2, 7, 11, 15];

const target = 9;

The function should return [0, 1] because nums[0] + nums[1] = 2 + 7 = 9.

**Requirements:**

* Implement the solution in JavaScript.
* The solution should have a time complexity better than O(n^2).
* Include proper error handling for edge cases.

**Question 3: MongoDB Aggregation** Consider a MongoDB collection named sales with documents structured as follows:

json:

{

    "\_id": ObjectId("..."),

    "date": ISODate("2024-06-15T00:00:00Z"),

    "store": "Store A",

    "items": [

        {

            "name": "item1",

            "quantity": 5,

            "price": 10.0

        },

        {

            "name": "item2",

            "quantity": 3,

            "price": 20.0

        }

    ]

}

Your task is to write an aggregation pipeline to calculate the total revenue generated by each store for each month, along with the average price of items sold. The result should be sorted first by store and then by month (in ascending order).

The expected output should be something like this:

json:

[

    {

        "store": "Store A",

        "month": "2024-06",

        "totalRevenue": 230.0,

        "averagePrice": 15.0

    },

    {

        "store": "Store B",

        "month": "2024-06",

        "totalRevenue": 150.0,

        "averagePrice": 12.5

    }

]

**Question 4: React** Build a Dynamic To-Do List Application in React.

**Objective:**

Create a React application that allows users to manage a to-do list with the following features:

* Add new tasks
* Mark tasks as complete
* Delete tasks
* Filter tasks by all, completed, and pending
* Persist tasks in the local storage so that they are retained after a page refresh

**Requirements:**

1. **UI Components:**
   * Create a TodoApp component as the main component.
   * Create a TodoList component to display the list of tasks.
   * Create a TodoItem component to represent each task.
   * Create an AddTodo component for adding new tasks.
   * Create a Filter component to filter tasks.
2. **State Management:**
   * Use the useState and useEffect hooks for state management.
   * Use local storage to persist the tasks.
3. **Functionality:**
   * Add a new task with a text input.
   * Mark a task as complete/incomplete by clicking on it.
   * Delete a task.
   * Filter tasks by all, completed, and pending.
   * Persist tasks in local storage.

PS - You may use this API for the todo app - [https://dummyjson.com/docs/todos#todos-a](https://dummyjson.com/docs/todos#todos-all)