Senior Full-stack Developer Assessment Task

Logic

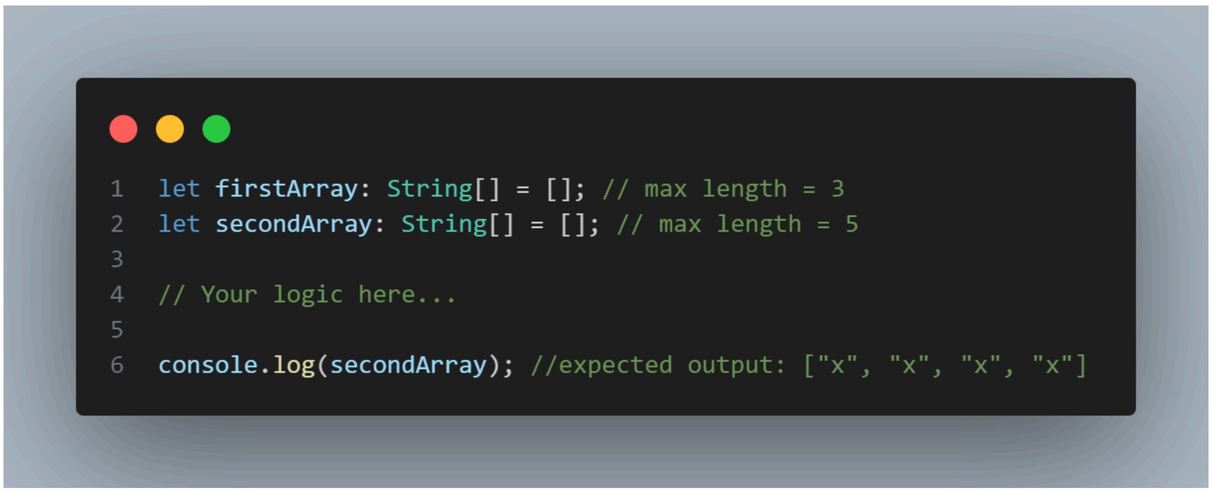
1. There are three light switches in the room where you are standing. In the adjacent room, three incandescent light bulbs are controlled by each of these switches. You are informed that all of the light bulbs in the other room are off, and all of the switches are originally down and in the off position.

   How can you figure out which switches match which light bulbs if you are only given **one chance** to enter the room with them?

2. You are given two arrays of string "x":
   - first array: can hold **at most 3 elements**:
   - second array: can hold **at most 5 elements**

   Initially, both arrays are empty. You may perform the following operations:
   - Fill an array to its **maximum** capacity (e.g., fill a second array with 5 items).
   - Empty an array
   - Move elements from one array to another until either:
     a. The **source** array is **empty**, or
     b. The **destination** array is **full**.

   Write a series of operations to measure exactly 4 elements in the second array, using only the operations above.

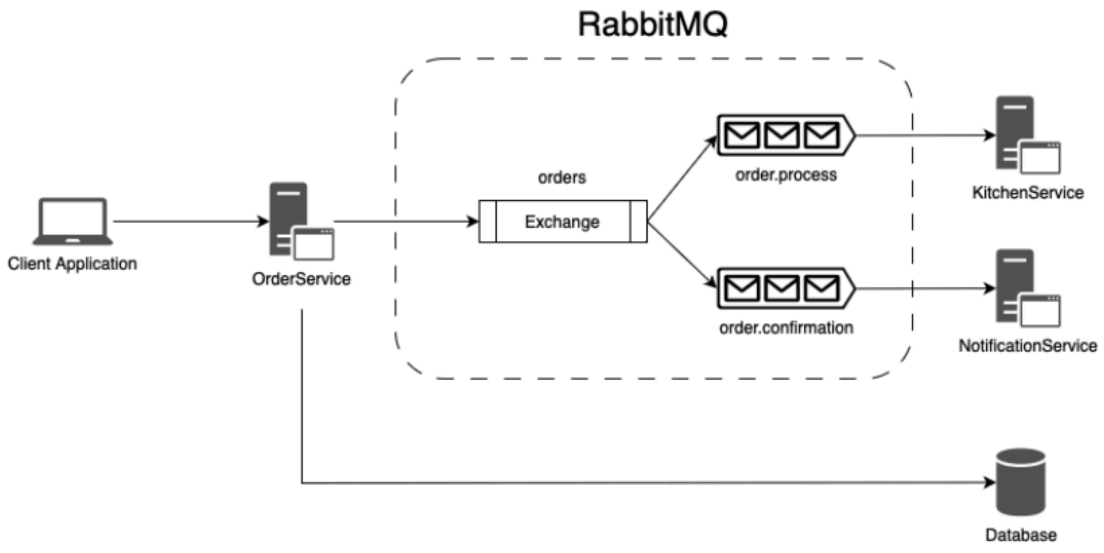


```
let firstArray: String[] = []; // max length = 3
let secondArray: String[] = []; // max length = 5

// Your logic here...

console.log(secondArray); //expected output: ["x", "x", "x", "x"]
```

Technical

Create a simple restaurant system with the following details:

- Use micro service architecture, with the following services
    a. OrderService
    b. KitchenService
    c. NotificationService
- Architecture reference:



- OrderService has REST endpoints that allow the user to:
    a. Fetch food menu
    b. Place an order
    c. Track the status of the order
- A food menu consists of name and price.
- An order consists of one or more food menu, a customer email, and order status.
- Once an order is placed, the order details will be recorded into a database, published into a message broker exchange, and it will return the Order ID to be used to track the order. The order status in this state will be "**Pending**".
- The message broker exchange is configured with "**fan-out**" pattern which will publish the order into 2 queues:
    a. **order.confirmation**, which will be consumed by **NotificationService** to send an email containing the order details to the user
    b. **order.process**, which will be consumed by **KitchenService** and updates the order status into "**Processed**"

Requirements:

- Use Docker-Compose so that the system / micro services can be deployed locally (target both X86 / X64 and ARM64 platform)
- Use NodeJS (minimum version 16) with NestJS framework
- Use RabbitMQ as the message broker
- Provide a database, preferably MySQL or MongoDB
- Pre-populate the database with some food menu data
- Use appropriate HTTP codes as responses for various scenario of the REST API endpoints, e.g: HTTP 200 OK, 401 Unauthorized, etc
- The business logic and object properties (account and other data properties etc) assumptions to be determined by candidate
- Ensure that the Docker containers can be deployed cross-platform (at least on Windows and Linux)
- Provide documentation to explain how to deploy the system
- Provide documentation to explain the functions developed in the code and how to test each features
- ZIP the codes or use GitHub to distribute the code