Question 1:
Write an Angular service method that fetches a list of products from an API endpoint (https://fakestoreapi.com/products). Use RxJS operators pipe() and map() to transform the response so that it only returns an array of product names.

Question 2:

Write an example of how to create an Observable that emits the values 1, 2, 3 with a 1-second delay between each emission. Subscribe to it and log the values to the console.

Question 3:

Create an Angular component that displays a list of product names using the ProductService (from question 1). Fetch the data inside ngOnInit() and display it in the template.

Question 4:

Write an Angular component that includes an input field where users can type their name. The name should be displayed in real-time below the input field using two-way data binding.

Question 5:

Create an Angular service that uses an RxJS Subject to allow components to share a message. Write a method to send messages and another to listen for messages.

Question 6:

The following RxJS implementation is supposed to filter out discounted products, but it's broken.

```
this.productService.getProducts()
  .pipe(
    map(products => products.filter(product => product.discount > 0)) // ❌ ERROR
  )
  .subscribe(filteredProducts => this.discountedProducts = filteredProducts);
```

Task:

1. **Fix the error** (Ensure that product.discount exists).
2. **Modify the map() function to filter correctly.**

- Check if product.discount is defined
- Ensure map() filters correctly

Question 7:

Write an Angular HttpInterceptor that attaches a Bearer token to all outgoing HTTP requests. The token should be retrieved from localStorage.

Question 8:

Create a simple Reactive Form in Angular with a single input field for email and a submit button. The form should be validated to check if the email input is not empty and follows a valid email format.

Question 9:

When the user selects a **theme**, it should update the UI dynamically.

Task:

1. Bind selectedTheme to a <select> dropdown.
2. When the user selects a theme, update the variable.

```
<label for="theme">Choose Theme:</label>
<select [(ngModel)]="selectedTheme">
  <option *ngFor="let theme of themes" [value]="theme">{{ theme }}</option>
</select>

<p>Selected Theme: {{ selectedTheme }}</p>
```

- Use [(ngModel)] for two-way binding
- Ensure the UI updates when the user selects a theme

Question 10:

The following **NavigationService** does not update the component when the navigation menu is updated.

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class NavigationService {
  private menuItems = new Subject<string[]>();

  addMenuItem(item: string) {
    this.menuItems.value.push(item); // ❌ ERROR: 'value' is not correct
  }
}
```

Task:

1. **Fix the error** in addMenuItem().
2. **Ensure the new menu item updates the observable correctly.**

- Use next() instead of value.push()
- Make sure menuItems emits updated values

Question 11:

The **product filter** component should allow users to **filter products by category**, but **it's not working**.

```
filterProducts(category: string) {
  return this.products.filter(p => p.category = category); // ❌ ERROR
}
```

Task:

1. Fix the filter() condition to compare correctly.
2. Ensure that the filtered products are updated in the UI.

- Use === (strict equality operator) instead of = in the filter condition
- Ensure the filter method correctly updates the product list

Question 12:

The user should be able to **subscribe to navigation updates** when they add a new menu item.

Task:

1. Implement a NavigationService using BehaviorSubject.
2. Ensure that components **subscribe** to navigation updates dynamically.

- **Use** BehaviorSubject **instead of** Subject
- **Ensure navigation updates are reflected in real-time**