# Problem 1: Indian Thali Builder

**Scenario:** You are creating an app for a restaurant that allows customers to build their own Indian thali (meal). Each thali can have a combination of items such as sabzi, dal, roti, rice, salad, and dessert.

**Task:**

1. Create a function `createThali` that takes multiple parameters: sabzi, dal, roti, rice, salad, and dessert.
2. Use default parameters to set a default thali if the user doesn't specify all items.
3. Use destructuring and the spread operator to allow customization of items in the thali.
4. Write a function `calculateCalories` that calculates the total calories of the thali based on a given set of items and their calories per serving.
5. Print out the details of the thali using template strings.

**Example Input:**

```
createThali({
  sabzi: "Paneer Butter Masala",
  dal: "Dal Tadka",
  roti: 2,
  rice: "Jeera Rice",
  salad: "Kachumber",
  dessert: "Gulab Jamun"
});
```

**Expected Output:**

```
Your Thali:
Sabzi: Paneer Butter Masala
Dal: Dal Tadka
Roti: 2
Rice: Jeera Rice
Salad: Kachumber
Dessert: Gulab Jamun
Total Calories: 1200
```

---

# Problem 2: Veggie Market Cart

**Scenario:** You're building a shopping cart system for a local online veggie market. The cart should keep track of the vegetables added, the quantity, and the total cost.

**Task:**

1. Create an object `veggiePrices` with prices per kg for different vegetables like `Potato`, `Tomato`, `Onion`, `Carrot`, etc.
2. Implement a `cart` object with methods to add items, remove items, and calculate the total cost.
3. Use object methods, `this` keyword, and optional chaining to safely access and update the cart.
4. Handle edge cases like adding an unavailable vegetable or removing a vegetable that isn't in the cart.

**Example Usage:**

```
cart.addItem("Potato", 2); // Adds 2 kg of Potato
cart.addItem("Tomato", 1.5); // Adds 1.5 kg of Tomato
cart.removeItem("Onion"); // Tries to remove Onion but it's not in the cart
cart.getTotal(); // Calculates and returns the total cost
```

**Expected Output:**

```
Cart: { Potato: 2, Tomato: 1.5 }
Total Cost: ₹150
```

---

# Problem 3: Spice Level Filter

**Scenario:** A food delivery service wants to provide users with options to filter vegetarian dishes based on spice levels (Mild, Medium, Spicy).

**Task:**

1. Create an array of objects representing different vegetarian dishes, each with a name, ingredients, spice level, and price.
2. Implement a function `filterBySpiceLevel` that takes the spice level as input and returns a list of dishes matching that spice level.
3. Use array methods like `filter`, `map`, and `forEach` to manipulate and display the dishes.
4. Sort the filtered dishes by price and print them in ascending order.

**Example Input:**

```
filterBySpiceLevel("Medium");
```

**Expected Output:**

```
Medium Spice Level Dishes:
1. Paneer Tikka - ₹200
2. Veg Biryani - ₹150
3. Dal Makhani - ₹120
```

---

# Problem 4: Samosa Stall Inventory

**Scenario:** You are managing an inventory system for a street-side samosa stall. The stall sells different types of samosas: Aloo, Paneer, Corn, and Spinach.

**Task:**

1. Create an object representing the inventory with keys as samosa types and values as quantities available.
2. Implement functions to add stock, sell samosas, and check stock levels.
3. Use `if-else` conditions and ternary operators to handle different scenarios such as insufficient stock or an invalid samosa type.
4. Use `for...in` loop to display the current stock.

**Example Usage:**

```
sellSamosa("Aloo", 10); // Sells 10 Aloo samosas
addStock("Paneer", 20); // Adds 20 Paneer samosas
checkStock("Corn"); // Checks stock level of Corn samosas
```

**Expected Output:**

```
Sold 10 Aloo Samosas
Added 20 Paneer Samosas
Corn Samosa Stock: 15
```

---

# Problem 5: Ladoo Production Line

**Scenario:** You are managing a ladoo production line where ladoos are made from different ingredients like besan, coconut, and atta. Each type of ladoo has a different production time and cost.

**Task:**

1. Create an array of ladoo objects with properties: type, ingredients, productionTime (in minutes), and cost.
2. Implement a function `startProduction` that simulates the production process using a `for` loop or `while` loop, printing the progress.
3. Use `setTimeout` or `setInterval` (for asynchronous practice) to simulate production time delays.
4. Calculate total production cost and time for a batch of ladoos.
5. Use a switch statement to handle different production processes based on ladoo type.

**Example Input:**

```
startProduction([{ type: "Besan", quantity: 10 }, { type: "Coconut", quantity: 5
}]);
```

**Expected Output:**

```
Producing 10 Besan Ladoos...
Done in 20 minutes. Cost: ₹100
Producing 5 Coconut Ladoos...
Done in 10 minutes. Cost: ₹75
Total Production Time: 30 minutes
Total Cost: ₹175
```

---