**Experiment no 2:**

**Problem statement 1: Array**

Write a Swift program to simulate a simple **music playlist manager** that uses an array to store song names as strings. The program should allow you to:

1. **Store Song Names**: Initialize an array of strings with the names of a few songs already present in the playlist.
2. **Add a New Song**: Add a new song to the playlist by appending it to the array.
3. **Remove a Song**: Remove a song from the playlist by its name. If the song does not exist, display an appropriate message.
4. **Display Songs in Alphabetical Order**: Sort the playlist in alphabetical order and display all the songs.

**Constraints:**

* Avoid using **loops**, **classes**, or **functions**.
* Use array operations like append, remove(at:), sort, and contains

**Problem Statement 2: Tuple**

Write a Swift program to compare three products using tuples. Each product will be represented by a tuple containing the following information:

1. **Product Name (String): The name of the product.**
2. **Price (Double): The price of the product.**
3. **Rating (Double): The rating of the product on a scale of 1 to 5.**

**Tasks to Perform:**

1. **Identify the Most Expensive Product:**
   * Compare the prices of the three products and determine the one with the highest price.
2. **Identify the Highest-Rated Product:**
   * Compare the ratings of the three products and determine the one with the highest rating.
3. **Find a Product with a Price Below a Given Threshold:**
   * Check if any of the products have a price below a given threshold (e.g., ₹500). If found, print the details of such a product. If no product meets this criterion, print "No products found under the given price threshold."
4. **Print the Details**:
   * Display the name, price, and rating of:
     + The most expensive product.
     + The highest-rated product.
     + The product below the threshold (if applicable).

**Problem statement 3: Set**

Write a Swift program to manage the attendance records of two workshops using **sets**. Each workshop has a list of attendees, and you need to perform specific operations on these sets to extract meaningful attendance data.

**Problem Requirements:**

1. **Input Data**:
   * Create two sets to store the names of attendees for **Workshop A** and **Workshop B**. Each attendee's name is a string.
2. **Operations to Perform**:
   * **Find attendees who attended both workshops**: Determine the intersection of the two sets to find common attendees.
   * **Find attendees who attended only one workshop**: Determine attendees exclusive to either Workshop A or Workshop B (no overlap).
   * **Combine all attendees into a single list without duplicates**: Find the union of the two sets to list all unique attendees.
3. **Output**:
   * Print the list of attendees who attended both workshops.
   * Print the list of attendees who attended only one workshop.
   * Print the combined list of all unique attendees from both workshops.

**Problem statement 4: Dictionary**

You need to create a Swift program that simulates the management of a grocery store's inventory. The inventory will be represented using a **dictionary**, where each key-value pair consists of:

1. **Key**: The name of an item (String).
2. **Value**: The stock quantity of the item (Int).

The program should perform the following operations:

**Functional Requirements:**

1. **Add a New Item to the Inventory**:
   * Allow the addition of a new item with a specified stock quantity.
   * If the item already exists, print an appropriate message and do not overwrite its stock.
2. **Update the Stock of an Existing Item**:
   * Update the stock quantity of an item already present in the inventory.
   * If the item does not exist, print a message indicating that the item is not found.
3. **Remove an Item from the Inventory**:
   * Remove an item by its name. If the item does not exist, print an appropriate message.
4. **Print All Items and Their Stock Quantities**:
   * Display the current inventory in a user-friendly format, showing each item's name and its stock quantity.