**Day 1: Core Java Fundamentals & OOP Concepts**

1. **Reverse a String**
   * Write a Java program to reverse a given string without using built-in methods.
   * Input: "Automation" → Output: "noitamotuA"
2. **Find the Duplicate Characters in a String**
   * Given a string, identify duplicate characters and their count.
   * Input: "Java Programming" → Output: {a=3, g=2, r=2}
3. **Palindrome Check**
   * Implement a function to check if a string or number is a palindrome.
   * Input: "madam", 12321 → Output: true
4. **OOP Design: Bank Account**
   * Implement a BankAccount class with features such as deposit, withdraw, and balance check using encapsulation and constructors.

**Day 2: Collections Framework**

1. **Find the First Non-Repeating Character**
   * Given a string, find the first non-repeating character using HashMap.
   * Input: "automation" → Output: 'u'
2. **Sort a List of Employee Objects**
   * Create an Employee class with id, name, salary fields, and sort employees by salary in descending order using Comparator.
3. **Remove Duplicates from an ArrayList**
   * Given an ArrayList of integers, remove duplicate elements efficiently.
4. **Merge Two Sorted Arrays Without Duplicates**
   * Merge two sorted arrays into one sorted array without duplicates.
   * Input: [1, 3, 5], [2, 3, 6] → Output: [1, 2, 3, 5, 6]

**Day 3: Exception Handling & File Handling**

1. **Custom Exception Handling**
   * Create a custom exception InvalidAgeException and throw it if age is below 18.
2. **Read and Write to a File**

* Write a Java program to read a file line by line and write content to another file.

1. **Count Words in a File**

* Read a text file and count occurrences of each word using Map.

**Day 4: Multithreading & Synchronization**

1. **Print Numbers Using Two Threads**

* Create two threads: one prints even numbers, the other prints odd numbers in sequence.

1. **Thread Synchronization Example**

* Implement a shared resource (e.g., bank account withdrawal) using synchronized methods.

1. **Producer-Consumer Problem**

* Implement the producer-consumer problem using wait() and notify().

**Day 5: Selenium WebDriver Hands-on**

1. **Automate Google Search**

* Write a Selenium script to open Google, search for a keyword, and print the first result title.

1. **Form Automation**

* Automate a sample form (fill text, radio, checkbox, dropdown, and submit the form).

1. **Screenshot Capture**

* Take a screenshot of a webpage using Selenium WebDriver and save it.

1. **Handling Dynamic Elements**

* Write code to interact with elements that load dynamically using Explicit Waits.

**Day 6: API Automation with Java**

1. **REST API Automation using RestAssured**

* Write a test to perform a GET request and validate the response status code and body.

1. **POST Request Automation**

* Send a POST request with JSON payload and validate the response.

1. **Data-Driven API Testing**

* Read test data from an external file (CSV/JSON) and automate API tests using a loop.

**Day 7: Framework Design & Integration**

1. **Design a Simple Automation Framework**

* Design a basic Selenium framework using TestNG with Page Object Model.

1. **Logging with Log4j**

* Integrate Log4j to generate logs for your test execution.

1. **Data-Driven Testing with Apache POI**

* Read data from Excel using Apache POI and use it in Selenium tests.