# GIRI's Tech Hub, Pune.

## Programming Machine Test

**Batch: Nov-23 to May-24 & Aug/Sep/Oct-24**

**Date : 23/06/2025**
**Time : 02 to 05 Pm.**

**Instructions:**

1. **Solve any 9 questions.**
2. **Input should be from user.**
3. **Indentation and comments mandatory.**

**Q1. Write a java program to print following pattern.**

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

**Q2. Problem Statement:**
**Create a Java application to manage product batches for a manufacturing company. Each batch contains multiple products. You must implement the following:**

**Requirements:**

1. **POJO Class:**
   **Create a Product class with the following fields:**
   - **productId (int)**
   - **productName (String)**
   - **unitPrice (double)**
   - **quantity (int)**
   - **batchId (int)**

   **Include a parameterized constructor, getter & setter methods, and a method double totalPrice() that returns the total price (unitPrice * quantity).**

2. **Main Operations in Main Class (do not use collections — use arrays only):**
   **Menu Options:**
   1. **Add Product**
   2. **Remove Product by ID**
   3. **Display All Products**
   4. **Find Batch with Highest Total Value**
   5. **Sort Products by Unit Price (Descending)**
   6. **Exit**

**Q3.Design and implement a Java program for a simple NumberProcessor system using inheritance that performs the following tasks on an integer array.**

**Specifications :**

1. **Create a base class NumberProcessor:**
   - **It should have an integer array numbers and an integer size.**
   - **It should have a constructor to initialize the array and its size.**
   - **It should have a method displayNumbers() to display all elements in the array.**

2. **Create a derived class NumberOperations that extends NumberProcessor :**
    - **findPrimeNumbers(): Display all prime numbers in the array.**
3. **Create a derived class NumberOperations that extends NumberProcessor :**
    - **findSecondLargest(): Find and display the second largest element in the array.**
4. **In the main method:**
    - **Ask the user to enter the size of the array (maximum 10).**
    - **Ask the user to input the array elements.**
    - **Create an object of NumberOperations and invoke all the above methods in a proper sequence.**

**Q4. Design and implement a Java program using Abstraction to perform advanced String operations without using Collections (like ArrayList, HashMap etc.).**

**Create an abstract class StringProcessor with the following abstract methods:**

1. **String findMaxOccurringChar(String input)**
2. **String reverseWords(String input)**

**Then, create a concrete class MyStringProcessor that extends StringProcessor and implements the logic for each method.**

**Requirements:**

- **findMaxOccurringChar: Find the character that appears the maximum number of times in the string and return it as a string. If multiple characters have the same max frequency, return the first one found.**
  ***Example:* input = "ababc" → output = "a"**
- **reverseWords: Reverse the order of words in a sentence, keeping each word intact. Words are separated by spaces.**
  ***Example:* input = "Java is awesome" → output = "awesome is Java"**

**Q5. Create a Java program using an interface to define methods for encoding and decoding strings.**

**Your task is to:**

1. **Create an interface StringCipher with two methods:**
    - **String encode(String input);**
    - **String decode(String input);**
2. **Create a class CustomCipher that implements StringCipher.**
    - **The encode method should perform the following:**
        - **For each character in the string:**
            - **If it's an alphabet (A-Z or a-z), shift it by 3 positions forward in the alphabet (like a Caesar cipher). Wrap around if needed (e.g., 'Z' -> 'C').**
            - **If it's a digit (0-9), replace it with its complement to 9 (e.g., 0→9, 1→8, 2→7, ..., 9→0).**
            - **Other characters should be left unchanged.**
    - **The decode method should reverse the above process.**
3. **Do not use any Java Collection classes. Use only primitive types, arrays, and strings.**
4. **Write a Main class with main method:**
    - **Take any input string (hardcode or use Scanner).**
    - **Call the encode method and print the encoded string.**
    - **Call the decode method on the encoded string and print the decoded result.**
    - **The decoded result must match the original input exactly.**

**Q6. You are required to create a Java program that performs addition and multiplication operations on elements of an integer array using two different threads. Use synchronization to ensure that only one thread at a time can access the array elements to avoid race conditions.**

**Requirements:**

1. **Create a class named ArrayOperator that has:**
   - **An integer array of size 5 (you can take any 5 numbers as input).**
   - **Two synchronized methods:**
     - **addElements() → adds 5 to each element of the array and prints the updated array.**
     - **multiplyElements() → multiplies each element by 2 and prints the updated array.**
2. **Create two thread classes:**
   - **AdditionThread → calls the addElements() method.**
   - **MultiplicationThread → calls the multiplyElements() method.**
3. **In the main method:**
   - **Initialize the array with any 5 numbers.**
   - **Start both threads.**
   - **Ensure that both operations are performed safely using synchronized methods.**

**Q7. Employee Management System — ArrayList Implementation**

**Problem Statement:**

**You are required to build a simple Employee Management System using ArrayList in Java.**

**Implement the following functionalities:**

1. **Add Employee — Add a new employee to the list. Each employee should have:**
   - **id (int)**
   - **name (String)**
   - **department (String)**
   - **salary (double)**
2. **View All Employees — Display the list of all employees.**
3. **Search Employee by ID — Display employee details for a given ID.**
4. **Update Employee Salary by ID — Update the salary of an employee using their ID.**
5. **Delete Employee by ID — Remove an employee using their ID.**
6. **Display Employees with Salary Greater than a Given Amount — Take an amount as input and display employees who earn more than that amount.**

**Constraints:**

- **Use an ArrayList to store Employee objects.**
- **Use proper OOP concepts (create an Employee class with appropriate fields and constructor).**
- **Use a menu-driven approach in the main program.**

**Q8. You are developing a small Food Management System for a cafeteria. You have to store the calories of different food items in a Vector<Integer>.**

**Requirements:**

1. **Input:**
   **-Store calories of at least 8 food items in a Vector<Integer>. (You can hardcode them.)**
2. **Tasks:**
   **a) Find and print the highest and lowest calorie food item.**
   **b) Calculate and print the average calories of all food items.**
   **c) Print all food items whose calories are above the average.**
   **d) Create a new Vector containing only even calorie values from the original Vector and display it.**
   **e) Count how many food items have calories in the range 200 to 500 (inclusive) and print the count.**

**Expected Output:**

**Calories list: [120, 350, 500, 450, 700, 250, 300, 400]**

**Highest calorie item: 700**

**Lowest calorie item: 120**

**Average calories: 383.75**
**Food items above average: [450, 500, 700, 400]**
**Even calorie food items: [120, 350, 500, 700, 250, 300, 400]**
**Food items in the range 200-500: 6**

**Q9. Problem Statement:**
**You are given a list of strings and a target string. Write a Java program that does the following:**

1. **Find all the strings from the list that are anagrams of the target string.**
2. **For all anagram strings found, count the frequency of each unique character across all the anagrams combined.**

**Input:**

- **A list of strings (e.g., ["listen", "silent", "enlist", "google", "inlets", "banana"])**
- **A target string (e.g., "listen")**

**Output:**

- **A list of anagram strings found.**

**Example :**
**Input : List of Strings: ["listen", "silent", "enlist", "google", "inlets", "banana"]**
**Target String: "listen"**
**Output : Anagrams found: [listen, silent, enlist, inlets]**

**Q10. Write a Java program that performs the following tasks:**

1. **Accept a text file named data.txt which contains multiple lines. Each line has a student's name followed by their marks in 3 subjects, separated by spaces.**
   **Example:**
   **John 78 85 90**
   **Alice 67 72 88**
   **Bob 90 92 85**
2. **Read the file and calculate the average marks for each student.**
3. **Write the student names and their average marks to a new file named result.txt, with each line in this format:**
   **John 84.33**
   **Alice 75.67**
   **Bob 89.00**

**-----ALL THE BEST-----**