The Test is divided into three sections. The first section has MCQ's on Java, each question carries 2 marks, And the second section has Queries on MySQL, each question carries 3 marks. For the third section, three questions need to be completed, each question carries 25 marks.

Duration of the Test: 3 hrs.

Section-1: Core Java MCQ's

1. **What is the output of the following code?**

```
public static void parse(String str) {
try {
float f = Float.parseFloat(str);
} catch (NumberFormatException nfe) {
f = 0;
} finally {
System.out.println(f);
}
}
public static void main(String[] args) {
parse("invalid");
}
```

A. 0.0

B. Compilation fails.

C. A ParseException is thrown by the parse method at runtime.

D. A NumberFormatException is thrown by the parse method at runtime.

2. **What is the output of the following code?**

```
class Employee {
String name; double baseSalary;
Employee(String name, double baseSalary) {
this.name = name;
this.baseSalary = baseSalary;
}
}
public class SalesPerson extends Employee {
double commission;
public SalesPerson(String name, double baseSalary, double commission) {
// insert code here
}
}
```

Which two code fragments, inserted independently at ("//insert code here"), will compile? (Choose two.)

A. super(name, baseSalary);

B. this.commission = commission;

C. super(); this.commission = commission;

D. this.commission = commission; super();

E. super(name, baseSalary); this.commission = commission;

F. this.commission = commission; super(name, baseSalary);

G. super(name, baseSalary, commission);

3. **Consider the following code:**

```
class Employee {
String name; double baseSalary;
Employee(String name, double baseSalary) {
this.name = name;
this.baseSalary = baseSalary;
}
}
public class SalesPerson extends Employee {
double commission;
public SalesPerson(String name, double baseSalary, double commission) {
// insert code here
}
}
```

Which two code fragments, inserted independently at ("//insert code here"), will compile? (Choose two.)

A. super(name, baseSalary);

B. this.commission = commission;

c. super(); this.commission = commission;

D. this.commission = commission; super();

E. super(name, baseSalary); this.commission = commission;

F. this.commission = commission; super(name, baseSalary);

G. super(name, baseSalary, commission);

4. **Consider the following code:**

```
class Pizza {
ArrayList toppings;
public final void addTopping(String topping) {
toppings.add(topping);
}
}
public class PepperoniPizza extends Pizza {
public void addTopping(String topping) {
System.out.println("Cannot add Toppings");
}
public static void main(String[] args) {
Pizza pizza = new PepperoniPizza();
pizza.addTopping("Mushrooms");
}
}
```

What is the result?

A. Compilation fails.

B. Cannot add Toppings

C. The code runs with no output.

D. A NullPointerException is thrown in Line 4.

5. **Consider the following code:**

```
public class Key {
private long id1;
private long id2;

// class Key methods
}
```

A programmer is developing a class Key, that will be used as a key in a standard java.util.HashMap. Which two methods should be overridden to assure that Key works correctly as a key? (Choose two.)

A. public int hashCode()

B. public boolean equals(Key k)

C. public int compareTo(Object o)

D. public boolean equals(Object o)

E. public boolean compareTo(Key k)

Section-2: MySQL Queries

Challenge 1: You have an employee database with a table called "employees" containing columns for employee ID, name, and manager ID.

Employees Table:

employee_id	name	manager_id
1	John	NULL
2	Alice	1
3	Bob	1
4	Carol	2
5	David	2
6	Eve	3
7	Frank	3

Query: Write a SQL query to retrieve a list of all employees and their direct managers' names.

Challenge 2: You have a product database with tables for "products" and "purchase_history."

Products Table:

product_id	product_name
1	Product A
2	Product B
3	Product C
4	Product D

Purchase History Table:

customer_id	product_id
101	1
102	1
102	2
103	1
103	2
103	3
104	1

Query: Write a SQL query to recommend products to customers based on their purchase history. Use subqueries to find products frequently purchased by customers who bought a specific item.

Challenge 3: You have a sales database with tables for "sales" and "territories."

Territories Table:

territory_id	territory_name
1	North
2	South
3	East
4	West

Sales Table:

sale_id	territory_id	amount
101	1	1000.00
102	1	1500.00
103	2	800.00
104	3	1200.00
105	4	700.00

Query: Write a SQL query to find the total sales for each territory and the average sales across all territories. Use joins to combine data from both tables.

Challenge 4: You have an inventory database with tables for "products," "suppliers," and "orders."

Products Table:

product_id	product_name
1	Product A
2	Product B
3	Product C
4	Product D

Suppliers Table:

supplier_id	supplier_name
101	Supplier X
102	Supplier Y
103	Supplier Z

Orders Table:

order_id	product_id	supplier_id	order_date
201	1	101	2022-01-15
202	2	102	2022-02-20
203	3	103	2022-03-10
204	1	102	2022-04-05
205	4	101	2022-05-15

Query: Write a SQL query to find the list of products, their suppliers, and the most recent order date for each product.

Challenge 5: You have a database for a school with tables for "students" and "courses."

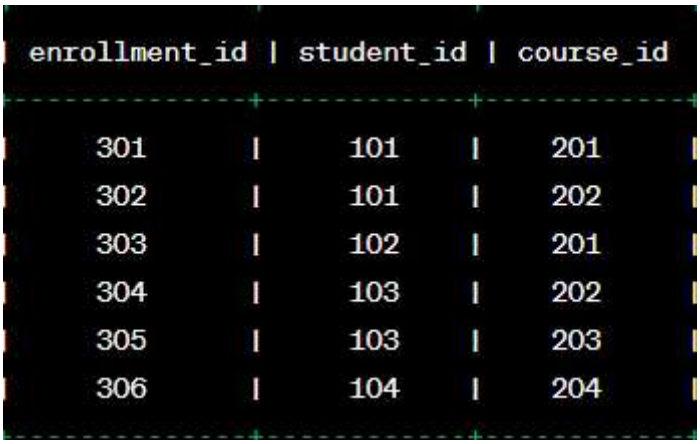
Students Table:

student_id	student_name
101	Alice
102	Bob
103	Carol
104	David

Courses Table:

course_id	course_name
201	Math 101
202	History 101
203	English 101
204	Science 101

Enrollments Table:



enrollment_id	student_id	course_id
301	101	201
302	101	202
303	102	201
304	103	202
305	103	203
306	104	204

Query: Write a SQL query to find the students who are enrolled in the most courses and list their names and the number of courses they are enrolled in. Use subqueries to determine the maximum number of course enrollments.

Note: You can use the provided table data to solve the SQL queries for each of the given challenge.

Section-3: Coding challenges on core java concepts

1. Write a Java program that determines if a given string can be formed by concatenating one or more words from a provided dictionary. The program should find and output all valid word combinations from the dictionary that can be used to form the input string. Words from the dictionary can be used multiple times, and the order of concatenation matters.

   Your program should provide the following functionality:

   Accept an input string from the user.

   Accept a dictionary of words (an array or list of words) from the user.

   Find and output all valid word combinations that can be used to form the input string.

   Sample input: iloveicecream

   Expected output: i love ice cream

2. Write a Java program that reads a list of words and finds all anagram groups within the list. An anagram group consists of words that can be formed by rearranging the letters of each other. Your program should identify and print all the anagram groups in the list.

   Your program should provide the following functionality:

   Accept a list of words as input.

   Identify and output all anagram groups found in the list.

   Sample input: ["listen", "silent", "hello", "world", "act", "cat"]

   Expected output:

   Anagram Group1: ["listen" and "silent"]

   Anagram Group2: ["act" and "cat"]

3. Write a java program to find the maximum sum subarray within a given array of integers. This array represents a time-series dataset and could be utilized for various applications, such as financial analysis, sensor data interpretation, or trend analysis. The maximum sum subarray is a contiguous subarray within the dataset that has the largest possible sum of its elements, potentially indicating a period of interest or significance in the data.

Design a robust and efficient solution to this problem. Your algorithm should be capable of handling large datasets and should return not only the maximum sum but also the starting and ending indices of the subarray.

Sample Input: [-2, -3, 4, -1, -2, 1, 5, -3]

Expected Output:

Maximum Sum Subarray: 7

4. Write a Java program to create a simplified e-commerce shopping cart system. The program should allow users to add products to their cart, remove products, view the cart's contents, and calculate the total cost. Implement exception handling for scenarios such as adding a product that doesn't exist, removing a product that's not in the cart, or negative quantities. Use custom exceptions for handling these scenarios.

   Your program should provide the following functionality:

- Create a Product class to represent products with attributes like ID, name, price, and quantity.
- Create a ShoppingCart class that allows users to:
- Add products to the cart.
- Remove products from the cart.
- View the cart's contents.
- Calculate the total cost of the items in the cart.

Implement custom exceptions (e.g., ProductNotFoundException, NegativeQuantityException) to handle errors gracefully.

Demonstrate the usage of your program by simulating a shopping experience where users interact with the cart and handle exceptions appropriately.

Sample input & output's:

Option 1: Add a Product to the Cart

Option 2: Remove a Product from the Cart

Option 3: View Cart Contents

Option 4: Calculate Total Cost

Option 5: Exit

Sample Input:

Choose option 1.
- Enter product ID: 1 (Laptop)
- Enter quantity: 2

Sample Output:

- Product "Laptop" (2 quantity) has been added to the cart.

Sample Input:

Choose option 2.
- Enter product ID: 2 (Phone)
- Enter quantity: 1

Sample Output:

- Product "Phone" (1 quantity) has been removed from the cart.

Sample Input:

Choose option 3.

Sample Output:

- Cart Contents:
- Laptop - Quantity: 2 - Price: $1000.0
- Phone - Quantity: 1 - Price: $500.0

Sample Input:

Choose option 4.
Sample Output:

- Total Cost: $2500.0

Sample Input:

Choose option 5.
Sample Output:

- Exiting the program.

Note: while displaying cart content or Or while inserting negative values or while removing if the user give invalid input handle the above mentioned exceptions there.

5. Write a Java program to create a simple task manager. The program should allow users to create, manage, and track tasks. Implement methods for creating, updating, and listing tasks, as well as marking tasks as completed.

   Your program should provide the following functionality:

   - Create a Task class to represent tasks with attributes like task title, description, due date, and completion status.
   - Create a TaskManager class that allows users to:
   - Create new tasks.
   - Update task details (title, description, due date, etc.).
   - Mark tasks as completed.
   - List all tasks (both completed and pending).

   Implement custom methods to handle complex task management operations efficiently.

   Demonstrate the usage of your program by allowing users to create, update, complete, and list tasks using a command-line interface.

   Sample input & output's:

   Option 1: Create task

   Option 2: Update status of task

   Option 3: Remove task

   Option 4: List all tasks

   Option 6: Exit

   Sample input:

   Choose option1

Create a task: "Task 1" with description "Description for Task 1" and due date "2023-11-01."
Create another task: "Task 2" with description "Description for Task 2" and due date "2023-11-15."

Sample output:

Task created successfully

Sample input:

Choose option2

Update the details of the first task: "Updated Task 1" title, "Updated description," and "2023-11-05" due date.
Mark the second task as completed.

Sample output:

Task updated successfully

Sample input:

Choose option3

Enter the title of the task: Task 1

Sample output:

Task 1 removed successfully.

Sample input:

Choose option4

Sample output:

Task List:

Task 1:

Title: Task 1

Description: Updated description

Due Date: 2023-11-05

Completed: false

Task 2:

Title: Task 2

Description: Description for Task 2

Due Date: 2023-11-15

Completed: true

Sample input:

Choose option5

Sample output:

Exiting the program