📝 Problem Statement: Advanced Shopping System with Discounts (Single Inheritance)

🎯 Objective:

Design and implement a C++ program that simulates a shopping system using **single inheritance**. The goal is to reinforce your understanding of class inheritance, constructor initialization, and parameter passing between parent and child classes.

📚 Background:

You are building a program for a shopping mall that manages **customer purchases**. The program must keep track of a customer's membership level, apply the appropriate discount, and calculate the total bill. Membership levels affect discount rates:

Membership Level	Discount Rate
Gold	20%
Silver	10%
Regular	5%

🧩 Class Design Requirements:

🔹 Class `Customer` (Parent Class)

- **Attributes:**
  - `string name`
  - `string membership`
  - `float discountRate`
- **Constructor:**
  - Accepts the customer's name and membership level.
  - Initializes the appropriate discount rate based on the membership.
  - Prints a message when the constructor is called.

🔹 Class `Purchase` (Child Class – inherits from `Customer`)

- **Attributes:**
  - `int itemCount`
  - `float pricePerItem`
  - `float totalAmount`
  - `float finalAmount`
- **Constructor:**

    - Accepts name, membership, item count, and price per item.
    - Passes the appropriate parameters to the parent class constructor.
    - Calculates:
        - `totalAmount = itemCount * pricePerItem`
        - `finalAmount = totalAmount - (discountRate * totalAmount)`

- **Method:**
    - `void displayInvoice()` – displays a formatted bill with all information.

☑ Expected Output Example:

```
Customer constructor called for Alice with membership: Gold
Purchase constructor called

--- Invoice ---
Customer: Alice
Membership: Gold
Items Bought: 5
Total Amount: $500
Discount Applied: 20%
Final Amount to Pay: $400
```

📌 Tasks:

1. Implement the above classes and logic as described.
2. Demonstrate inheritance and constructor chaining clearly in your code.
3. Format your output to display a clean invoice.
4. Add at least one additional validation (e.g., item count cannot be negative).
5. Include meaningful comments in your code.

✹ Bonus Challenge (Optional):

- Add a **tax calculation** (e.g., 5%) **after applying the discount**.
- Extend the `Customer` class to handle multiple purchase histories.
- Introduce different product categories with their own discount adjustments.

🔄 Submission Guidelines:

- Submit your `.cpp` file with meaningful class names and clean formatting.
- Include sample output in comments.
- Code should compile and run without errors.
