**Processing Orders with Validation**

**Problem Statement:** Processing Orders with Validation

**Objective**: Develop a console application that simulates an e-commerce system to process customer orders, using a multicast delegate to apply multiple validation rules (e.g., checking order amount and customer eligibility) before processing the order.

**Description**:

The application should model an e-commerce system where orders are validated against multiple criteria before being processed. A delegate will be used to define and combine validation rules, allowing flexible and extensible validation logic. The program should:

- Define an `Order` class with properties such as `Amount` and `CustomerId`.

- Use a delegate to represent validation rules that return a boolean indicating whether the order passes the rule.

- Implement a method to process an order, invoking all validation rules via a multicast delegate.

- Include at least two validation rules (e.g., ensuring a minimum order amount and validating the customer ID).

- Display the result of each validation step and whether the order was processed successfully or failed due to validation.

- Demonstrate the validation process with both valid and invalid orders.

**Requirements**:

1. Define a delegate type for validation methods that take an `Order` object and return a boolean.

2. Implement an `Order` class with at least `Amount` (decimal) and `CustomerId` (integer) properties.

3. Create a method to process an order, which invokes all validation rules using a multicast delegate and processes the order only if all rules pass.

4. Implement at least two validation rules (e.g., minimum order amount of $10, valid customer ID greater than 0) using named methods or lambda expressions.

5. Use a multicast delegate to combine multiple validation rules.

6. Display clear messages for each validation step and the final result (success or failure).

7. Test the application with at least one valid and one invalid order to demonstrate the validation logic.

**Expected Input:**

- Predefined `Order` objects (e.g., `Amount=50, CustomerId=101` for a valid order, and `Amount=5, CustomerId=101` for an invalid order).

- No direct user input is required; the program should demonstrate validation programmatically.

**Expected Output:**

For a valid order:

Validating order: Amount=50, CustomerId=101

Order amount is sufficient.

Customer ID is valid.

Order processed successfully.

For an invalid order:

Validating order: Amount=5, CustomerId=101

Order amount is too low.

Order validation failed.

**Constraints**:

- The application must be console-based and straightforward to understand.

- Use a multicast delegate to combine multiple validation rules.

- All validation rules must pass for the order to be processed successfully.

- Validation rules should be independent and reusable.

- The program should handle both valid and invalid cases clearly.

**Learning Goals:**

- Understand how to use multicast delegates to chain multiple validation rules in C#.

- Learn to handle return values from delegates in a multicast scenario.

- Gain experience with object-oriented programming (using the `Order` class) and delegate-based validation logic.

- Explore practical applications of delegates in business rule processing.