Exercise 1: The "Add & Remove" Pattern

Scenario: You are building a file system. Create an enum FileAttributes with ReadOnly (1), Hidden (2), System (4), and Archive (8).

- **Task:** Write a method ToggleArchive(FileAttributes current) that adds the Archive flag if it's missing, or removes it if it's already there.
- **Constraint:** Use the XOR (^) operator.

Exercise 2: The "Permission Guard"

Scenario: Create an enum UserRoles with Guest (1), Editor (2), and Admin (4).

- **Task:** Write a function CanEdit(UserRoles user) that returns true only if the user has the Editor flag **OR** the Admin flag.
- **Constraint:** Use a single if statement with bitwise OR (|) and AND (&) operators.

Exercise 3: Validation Logic

Scenario: You have a MachineStatus enum: PoweredOn (1), Running (2), Error (4), and MaintenanceMode (8).

- **Task:** Write a method IsValidState(MachineStatus status) that returns false if the machine is both Running and in MaintenanceMode at the same time.
- **Constraint:** Check the bits manually without using .HasFlag().

Exercise 4: The "Bulk Header"

C#

```
[Flags]
public enum CoffeeToppings {
    None = 0,
    Milk = 1,
    Sugar = 2,
    Caramel = 4,
    FullFlavor = Milk | Sugar | Caramel // The Combo
}
```

Scenario: Often, we create "Combo" flags within the enum itself for convenience.

- **Task:** Write a program that takes a variable myCoffee. If myCoffee has all three ingredients, print "Full House!". If it only has Milk and Sugar, print "Standard".
- **Constraint:** Use the FullFlavor named constant in your comparison logic.

Exercise 5: Binary Math (Power of Two)

Scenario: You are dynamically generating flags.

- **Task:** Create an enum NetworkProtocols. Instead of typing 1, 2, 4, 8, use the **bit-shift operator** (<<) to define the values up to the 5th bit (16).
- **Example format:** Protocols = 1 << 0,
- **Constraint:** Verify that the 5th protocol equals 16 using a Console.WriteLine.