Challenge 1: Interface Factory

Create a program that demonstrates the Factory Creational Pattern using an interface factory for shapes. You should have three types of shapes: Circle, Square, and Triangle.

1. Create an interface called `Shape` with a method `Area() float64`.
2. Implement three struct types: `Circle`, `Square`, and `Triangle`. Each should implement the `Shape` interface and have appropriate fields.
3. Finally, create a program that demonstrates how to use these factories to create shapes and calculate their areas, just like the code below.

```
 // Demonstrate using the factories to create shapes and calculate their areas

func main() {
    circle := NewShape(1, 0.5) // sides, radius
    fmt.Printf("Area of Circle: %.2f\n", circle.Area())
    // Same for triangle and rectangle
}
```

Challenge 2: Factory Generator

Create a Factory Generator that dynamically generates Shape Factories for Circle, Square, and Triangle. This will allow you to create instances of Shape Factories for any shape type without having to implement each factory manually.

1. Create a Factory Generator struct called `ShapeFactory` with a method `NewShapeFactory() ShapeFactory`.
2. Implement a concrete Factory Generator type that generates `ShapeFactory` instances for Circle, Square, and Triangle dynamically.
3. Modify your program to use the Factory Generator to create instances of Shape Factories for different shapes and then use those factories to create and calculate areas of the respective shapes.
4. Implement the functional and structural factory generators

Here's a simplified example to get you started:

```
 //  Use the Factory Generator to generate Shape Factories

func main() {
    circleGenerator := NewShapeFactory(1) // sides
    circle := circleGenerator("0.5") // radius
    fmt.Printf("Area of Circle: %.2f\n", circle.Area())
    // same for triangle and rectangle

}
```

In this challenge, you'll need to create a Factory Generator that can generate Shape Factories for different shapes. Modify the program accordingly and ensure that it works with dynamically generated factories.

Challenge 3: Prototype Factory

Create a Prototype Factory that uses prototypes of Circle, Square, and Triangle to create new instances of shapes. This should involve cloning existing shape objects to create new ones.

1. Implement a Prototype Factory struct called `ShapeFactory` and a `NewShape` function
2. Create concrete implementations of the Prototype Factory that store prototype instances for each shape type.
3. Modify your program to use the Prototype Factory to create new instances of shapes by cloning the prototypes. You should be able to clone and create multiple instances of each shape type.

Here's a simplified example of how the factory should be used:

```
 // Create prototypes for Circle, Square, and Triangle

func main() {
    circle := NewShape("circle")
    circle.radius = 0.5
    fmt.Printf("Area of Circle: %.2f\n", circle.Area())
}
```

In this challenge, you'll implement a Prototype Factory that allows you to create new instances of shapes by cloning the prototypes. Modify your program accordingly and ensure that it works with the Prototype Factory.