1. **Function Overloading:**
   - Define two overloaded functions named `calculateArea` that calculate the area of a rectangle and a circle. The first function should take two parameters (length and breadth) and return the area of a rectangle. The second function should take one parameter (radius) and return the area of a circle.
2. **Higher-Order Functions:**
   - Write a higher-order function named `operation` that takes two integers and a function as parameters. The function should apply the given function to the two integers and return the result.
3. **Extension Functions:**
   - Create an extension function for the `String` class named `isPalindrome` that returns true if the string is a palindrome, and false otherwise.
4. **Recursive Functions:**
   - Write a recursive function to calculate the factorial of a number. Optimize the function to avoid stack overflow for large inputs.

Advanced Programming Questions on Classes

1. **Inheritance and Polymorphism:**
   - Create an abstract class `Shape` with an abstract method `area`. Create two subclasses `Rectangle` and `Circle` that implement the `area` method. Add another method `perimeter` in both subclasses and demonstrate polymorphism.
2. **Data Classes and Copy Function:**
   - Define a data class `Person` with properties `name`, `age`, and `address`. Create an instance of `Person` and use the `copy` function to create a new instance with a different age and address.
3. **Companion Objects and Factory Methods:**
   - Create a class `User` with a companion object that has a factory method to create a user with a default age. Extend the factory method to allow creation of a user with optional parameters for name and age.
4. **Interface Implementation:**
   - Create an interface `Drawable` with a method `draw`. Implement this interface in two classes `Square` and `Triangle`. Extend the interface to include a method `resize` and demonstrate its usage in both classes.
5. **Singleton Pattern:**
   - Implement the Singleton pattern using Kotlin's object declaration. Create a class `DatabaseConnection` that ensures only one instance of the connection exists throughout the application.