**Question 1**

Delegation followed by Inheritance Inheritance followed by Delegation 

Fill the blanks with the correct methodology to emphasize the differences between the Adapter and the Bridge Pattern:

The Adapter Pattern is used to make unrelated components work together. Method: 

***[Inheritance followed by Delegation]***

The Bridge Pattern is used in a design to let abstractions and implementations vary independently.  Method: 

***[Delegation followed by Inheritance]***

Question 2: In which of the following scenarios would you use the Strategy Pattern?

Select one or more:

a. To switch implementation at run time

b. To provide an interface for a legacy class Incorrect

c. There are multiple algorithms for the same task

d. To grant access control to a resource

The correct answers are: There are multiple algorithms for the same task, to switch implementation at run time

**Question 3:** Assign the following Design Patterns to the categories used by Gamma.

Structural pattern, creational pattern, behavioral pattern

|  |
| --- |
| Bridge Pattern |
| Adapter Pattern |
| Proxy Pattern |
| Strategy Pattern |
| Abstract Factory Pattern |
| Facade Pattern |
| Observer Pattern |
| Composite Pattern |

The correct answer is: Bridge Pattern → Structural Patterns, Adapter Pattern → Structural Patterns, Proxy Pattern → Structural Patterns, Strategy Pattern → Behavioral Patterns,   
Abstract Factory Pattern → Creational Patterns, Facade Pattern → Structural Patterns, Observer Pattern → Behavioral Patterns, Composite Pattern → Structural Patterns

**Question 4:** Match the following features to the categorization used by Gamma:

Structural pattern, creational pattern, behavioral pattern

|  |
| --- |
| Reduce coupling between two or more classes |
| Introduce an abstract class to enable future extension |
| Model complex control flow |
| Make the system independent from the way its objects are created |
| Allow a choice between algorithms |
| Abstract from complex instantiation processes |

The correct answer is: Reduce coupling between two or more classes → Structural Patterns, Introduce an abstract class to enable future extension → Structural Patterns, Model complex control flow → Behavioral Patterns, Make the system independent from the way its objects are created → Creational Patterns, Allow a choice between algorithms → Behavioral Patterns, Abstract from complex instantiation processes → Creational Patterns

**Question 5:** Match the following definitions:

Pattern, Algorithm

|  |
| --- |
| Starts from an initial state and proceeds through a series of successive states, eventually terminating in a final state: |
| A method for solving a problem using a finite sequence of well-defined instructions: |
| Describes a problem which occurs over and over again and describes the core of the solution to that problem: |
| Uses a solution a million times over, without ever doing it the same way twice: |

The correct answer is: Starts from an initial state and proceeds through a series of successive states, eventually terminating in a final state: → Algorithm, A method for solving a problem using a finite sequence of well-defined instructions: → Algorithm,   
Describes a problem which occurs over and over again and describes the core of the solution to that problem: → Pattern, Uses a solution a million times over, without ever doing it the same way twice: → Pattern