1:

a: Observer

b:

```
class Program
{
   static void Main()
   {
      var exam = new Exam("Matemáticas");
      var student1 = new Student("Alice");
      var student2 = new Student("Bob");

      exam.Subscribe(student1);
      exam.Subscribe(student2);

      exam.StartExam();
   }
}

interface IObserver
{
   void Update(string subject);
}

interface ISubject
{
   void Subscribe(IObserver observer);
   void Unsubscribe(IObserver observer);
   void Notify();
}

class Exam : ISubject
{
   private List<IObserver> observers = new List<IObserver>();
   public string Subject { get; }

   public Exam(string subject)
   {
      Subject = subject;
   }

   public void Subscribe(IObserver observer)
   {
      observers.Add(observer);
   }

   public void Unsubscribe(IObserver observer)
   {
      observers.Remove(observer);
   }

   public void Notify()
   {
      foreach (var observer in observers)
      {
```

```
            observer.Update(Subject);
        }
    }

    public void StartExam()
    {
        Console.WriteLine($"¡Comienza el examen de {Subject}!");

        // Realizar cualquier lógica necesaria antes del examen

        Notify();
    }
}

class Student : IObserver
{
    public string Name { get; }

    public Student(string name)
    {
        Name = name;
    }

    public void Update(string subject)
    {
        Console.WriteLine($"{Name}, hay un nuevo examen de {subject}!");
    }
}
```

2:

a: Se presenta una situación donde el estado de un objeto es guardado y vuelto a cargar repetidamente. Esto se puede implementar de mejor forma usando el patron Memento.

b:

```
public class GameCharacter
{
    public string Name { get; set; }
    public int Health { get; set; }
    public int Mana { get; set; }

    public void DisplayStatus()
    {
        Console.WriteLine($"{Name} tiene {Health} de salud y {Mana} de mana.");
    }

    public class Memento
    {
        private string Name { get; set; }
        private int Health { get; set; }
        private int Mana { get; set; }
    }

    public Memento SaveState()
    {
        return new GameCharMemento
        {
               Name = this.Name,
               Health = this.Health,
               Mana = this.Mana
        }
    }

    public void LoadState(Memento m)
    {
        this.Name = m.Name;
        this.Health = m.Health;
        this.Mana = m.Mana;
    }
}

public class Program
{
    public static void Main()
    {
        var gameCharacter = new GameCharacter
        {
               Name = "John",
               Health = 100,
```

```
            Mana = 50
        };

        Console.WriteLine("Estado inicial:");
        gameCharacter.DisplayStatus();

        Console.WriteLine("\nGuardando estado...");
        var savedState = gameCharacter.SaveState();

        Console.WriteLine("\nCambiando estados...");
        gameCharacter.Health -= 30;
        gameCharacter.Mana += 20;
        gameCharacter.DisplayStatus();

        Console.WriteLine("\nRestaurando estado...");
        gameCharacter.LoadState(savedState);
        gameCharacter.DisplayStatus();
    }
}
```

3:

a: Mediator

b:

```
class Program
{
    static void Main()
    {
        var alice = new User("Alice");
        var bob = new User("Bob");

        var IChatRoom = new ChatRoom();
        IChatRoom.SendMessage("Hola Bob!", bob, alice);
        IChatRoom.SendMessage("Hola Alice!", alice, bob);
    }
}

interface IUser
{
    string Name { get; }
}

interface IChatRoom
{
    void SendMessage(string message, IUser recipient, IUser sender);
}

class ChatRoom : IChatRoom
{
    public void SendMessage(string message, IUser recipient, IUser sender)
    {
        Console.WriteLine($"{sender.Name} to {recipient.Name}: {message}");
    }
}

class User
{
    public string Name { get; }
    public User(string name)
    {
        Name = name;
    }
}
```

4:

a: Command

b:

```
class Program
{
   static void Main()
   {
      var television = new Television();
      var remoteControl = new RemoteControl();

      remoteControl.SetCommand("on", new TurnOnCommand(television));
      remoteControl.SetCommand("off", new TurnOffCommand(television));
      remoteControl.SetCommand("volumeup", new VolumeUpCommand(television));
      remoteControl.SetCommand("volumedown", new VolumeDownCommand(television));

      string input = "";
      while (input != "exit")
      {
         Console.WriteLine("Escribe 'on' para encender, 'off' para apagar, 'volumeup' para subir volumen,
'volumedown' para bajar volumen, 'exit' para salir.");
         input = Console.ReadLine();
         remoteControl.ExecuteCommand(input);
      }
   }
}

interface ICommandl
{
   void Execute();
}

class RemoteControl
{
   private Dictionary<string, ICommand> commands = new Dictionary<string, ICommand>();

   public void SetCommand(string commandName, ICommand command)
   {
      commands[commandName] = command;
   }

   public void ExecuteCommand(string commandName)
   {
      if (commands.ContainsKey(commandName))
      {
         commands[commandName].Execute();
      }
      else
      {
         Console.WriteLine("Comando no reconocido.");
      }
   }
}

class TurnOnCommand : ICommand
{
```

```
    private readonly Television television;

    public TurnOnCommand(Television television)
    {
        this.television = television;
    }

    public void Execute()
    {
        television.TurnOn();
    }
}

class TurnOffCommand : ICommand
{
    private readonly Television television;

    public TurnOffCommand(Television television)
    {
        this.television = television;
    }

    public void Execute()
    {
        television.TurnOff();
    }
}

class VolumeUpCommand : ICommand
{
    private readonly Television television;

    public VolumeUpCommand(Television television)
    {
        this.television = television;
    }

    public void Execute()
    {
        television.VolumeUp();
    }
}

class VolumeDownCommand : ICommand
{
    private readonly Television television;

    public VolumeDownCommand(Television television)
    {
        this.television = television;
    }

    public void Execute()
    {
        television.VolumeDown();
    }
```

```
}

class Television
{
    private bool isOn = false;
    private int volume = 10;

    public void TurnOn()
    {
        isOn = true;
        Console.WriteLine("Televisión encendida.");
    }

    public void TurnOff()
    {
        isOn = false;
        Console.WriteLine("Televisión apagada.");
    }

    public void VolumeUp()
    {
        if (isOn)
        {
            volume++;
            Console.WriteLine($"Volumen: {volume}");
        }
    }

    public void VolumeDown()
    {
        if (isOn)
        {
            volume--;
            Console.WriteLine($"Volumen: {volume}");
        }
    }
}
```

5:

a: Visitor

b:

```
interface IAnimalVisitor
{
    public void VisitLion(Lion lion);
    public void VisitMonkey(Monkey monkey);
    public void VisitElephant(Elephant elephant);
}

class AnimalVisitor : IAnimalVisitor
{
    public void VisitLion(Lion lion)
    {
        lion.Feed();
    }

    public void VisitMonkey(Monkey monkey)
    {
        monkey.Feed();
    }

    public void VisitElephant(Elephant elephant)
    {
        elephant.Feed();
    }

}

abstract class Animal
{
    public abstract void Feed();
    public abstract void AcceptVisitor(IAnimalVisitor visitor);
}

class Lion : Animal
{
    public override void Feed()
    {
        Console.WriteLine("El león está siendo alimentado con carne.");
    }

    public override void AcceptVisitor(IAnimalVisitor visitor)
    {
        visitor.VisitLion(this);
    }
}

class Monkey : Animal
```

```
{
    public override void Feed()
    {
        Console.WriteLine("El mono está siendo alimentado con bananas.");
    }

    public override void AcceptVisitor(IAnimalVisitor visitor)
    {
        visitor.VisitMonkey(this);
    }
}

class Elephant : Animal
{
    public override void Feed()
    {
        Console.WriteLine("El elefante está siendo alimentado con pastito.");
    }

    public override void AcceptVisitor(IAnimalVisitor visitor)
    {
        visitor.VisitElephant(this);
    }
}

class Program
{
    static void Main()
    {
        Animal[] animals = { new Lion(), new Monkey(), new Elephant() };

        IAnimalVisitor visitor = new AnimalVisitor();

        foreach (var animal in animals)
        {
               animal.Accept(visitor);
        }
    }
}
```

6:

a: State

b:

```
class Program
{
    static void Main()
    {
        var trafficLight = new TrafficLight();
        for (int i = 0; i < 5; i++)
        {
            trafficLight.ChangeLight();
            Thread.Sleep(1000); // Wait 1 second
        }
    }
}
class TrafficLight
{
    enum Light { Red, Yellow, Green }
    private ITrafficLightState state;
    public TrafficLight()
    {
        state = new RedLight();
        Console.WriteLine("Luz inicial es Roja.");
    }
    public void ChangeLight()
    {
        this.state.ChangeLight(this);
    }
}

interface ITrafficLightState
{
    public TrafficLight.Light CurrentLight { get; }
    void ChangeLight(TrafficLight light);
}

class RedLight : ITrafficLightState
{
    public TrafficLight.Light CurrentLight => TrafficLight.Light.Red;
    public void ChangeLight(TrafficLight light)
    {
        light.State = new GreenLight();
        Console.WriteLine("Cambio a Verde.");
    }
}

class GreenLight : ITrafficLightState
{
    public TrafficLight.Light CurrentLight => TrafficLight.Light.Green;
    public void ChangeLight(TrafficLight light)
    {
```

```
        light.State = new YellowLight();
        Console.WriteLine("Cambio a Amarillo.");
    }
}

class YellowLight : ITrafficLightState
{
    public TrafficLight.Light CurrentLight => TrafficLight.Light.Yellow;
    public void ChangeLight(TrafficLight light)
    {
        light.State = new RedLight();
        Console.WriteLine("Cambio a Rojo.");
    }
}
```

7:

a: Strategy

b:

```
public interface IShippingStrategy(){

	public double CalculateShippingCost(double weight);

}
public class ShoppingByUPS() : IShippingStrategy{

	public double CalculateShippingCost( double weight )
	{
		return weight * 0,75
	}

}
public class ShoppingByFedEx() : IShippingStrategy{

	public double CalculateShippingCost( double weight )
	{
		return weight * 0,85
	}

}
public class ShoppingByDAC() : IShippingStrategy{

	public double CalculateShippingCost( double weight )
	{
		return weight * 0,65
	}

}
class ShippingCalculator
{
	private IShippingStrategy shippingStrategy;

	public changeStrat( IShippingStrategy shippingStrat){
		this.shippingStrategy = shippingStrat
	}

	public double CalculateShippingCost( double weight )
	{
		if(shippingStrategy != null){
			this.shippingStrategy.CalculateShippingCost(weight)
		}else{
			throw new Exception("Courier no soportado.");
		}
	}
	class Program
	{
	  static void Main()
	  {
		var shippingCalculator = new ShippingCalculator();
```

```
                    shippingCalculator.SetShippingStrategy(new UPSShippingStrategy());
                    Console.WriteLine("Costo de envío con UPS: " +
shippingCalculator.CalculateShippingCost(5));

                    shippingCalculator.SetShippingStrategy(new FedExShippingStrategy());
                    Console.WriteLine("Costo de envío con FedEx: " +
shippingCalculator.CalculateShippingCost(5));

                    shippingCalculator.SetShippingStrategy(new DACShippingStrategy());
                    Console.WriteLine("Costo de envío con DAC: " +
shippingCalculator.CalculateShippingCost(5));
                }
            }

}
```

8:

a: Command

b:

```
class Program
{
    static void Main()
    {
       var emailService = new EmailService();

       Command emailCommand = new SendEmailCommand(emailService,
"john.doe@example.com", "Nueva promoción", "¡Revisa nuestra nueva promoción!");

       Command newsLetterCommand = new SendNewsLetterCommand(emailService,
"john.doe@example.com", "Newsletter de Junio", "Aquí está nuestro newsletter de
Junio.");

       emailCommand.Execute();
       newsLetterCommand.Execute();
    }
}

abstract class Command
{
    public EmailService Service { get; init; }

    public Command(EmailService service)
    {
        this.Service = service;
    }

    public abstract void Execute();
}

class SendEmailCommand : Command
{
    public string Recipient { get; set; }
    public string Subject { get; set; }
    public string Message { get; set; }

    public SendEmailCommand(EmailService service, string recipient, string subject,
string message) : base(service)
    {
        this.Recipient = recipient;
        this.Subject = subject;
        this.Message = message;
    }

    public override void Execute()
    {
```

```
        this.Service.SendEmail(this.Recipient, this.Subject, this.Message);
    }
}

class SendNewsLetterCommand : Command
{
    public string Recipient { get; set; }
    public string Subject { get; set; }
    public string Message { get; set; }

    public SendNewsLetterCommand(EmailService service, string recipient, string
subject, string message) : base(service)
    {
        this.Recipient = recipient;
        this.Subject = subject;
        this.Message = message;
    }

    public override void Execute()
    {
        this.Service.SendNewsLetter(this.Recipient, this.Subject, this.Message);
    }
}

class EmailService
{
    public void SendEmail(string recipient, string subject, string message)
    {
        Console.WriteLine($"Enviando correo a {recipient} con el asunto
'{subject}': {message}");
        // Agregar código para enviar correo
    }

    public void SendNewsletter(string recipient, string subject, string message)
    {
        Console.WriteLine($"Enviando newsletter a {recipient} con el asunto
'{subject}': {message}");
        // Agregar código para enviar newsletter
    }
}
```

9:

a: Chain of Responsibility

b:

```
class Program
{
    static void Main()
    {
        SupportSystem supportSystem = new SupportSystem();
        supportSystem.HandleSupportRequest(1, "No puedo iniciar sesión.");
        supportSystem.HandleSupportRequest(2, "Mi cuenta ha sidobloqueada.");

        supportSystem.HandleSupportRequest(3, "Necesito recuperar datosborrados.");
    }
}
class SupportSystem
{
    public SupportSystem()
    {
        _level1SupportRequestHandler = new Level1SupportRequestHandler();
        _level2SupportRequestHandler = new Level2SupportRequestHandler();
        _level3SupportRequestHandler = new Level3SupportRequestHandler();

        _level1SupportRequestHandler.SetNextHandler(_level2SupportRequestHandler);
        _level2SupportRequestHandler.SetNextHandler(_level3SupportRequestHandler);
    }

    public void HandleSupportRequest(int level, string message)
    {
        _level1SupportRequestHandler.HandleRequest(level, message);
    }
}

interface ISupportRequestHandler
{
    void SetNextHandler(ISupportRequestHandler nextHandler);
    void HandleRequest(int level, string message);
}

abstract class BaseSupportRequestHandler : ISupportRequestHandler
{
    protected ISupportRequestHandler _nextHandler;
    public void SetNextHandler(ISupportRequestHandler nextHandler)
    {
        _nextHandler = nextHandler;
    }

    public void HandleRequest(int level, string message)
    {
        if (CanHandleRequest(level))
```

```
        {
            HandleRequest(message);
        }
        else if (_nextHandler != null)
        {
            _nextHandler.HandleRequest(level, message);
        }
        else
        {
            Console.WriteLine("Consulta no soportada.");
        }
    }

    protected abstract void HandleRequest(string message);
    protected abstract int GetHandlerLevel();

    protected bool CanHandleRequest(int level)
    {
        return level == GetHandlerLevel();
    }
}

class Level1SupportRequestHandler : BaseSupportRequestHandler
{
    protected override void HandleRequest(string message)
    {
        Console.WriteLine("Soporte de Nivel 1: Manejando consulta - " + message);
    }

    protected override int GetHandlerLevel()
    {
        return 1;
    }
}

class Level2SupportRequestHandler : BaseSupportRequestHandler
{
    protected override void HandleRequest(string message)
    {
        Console.WriteLine("Soporte de Nivel 2: Manejando consulta - " + message);
    }

    protected override int GetHandlerLevel()
    {
        return 2;
    }
}

class Level3SupportRequestHandler : BaseSupportRequestHandler
{
    protected override void HandleRequest(string message)
```

```
    {
        Console.WriteLine("Soporte de Nivel 3: Manejando consulta - " + message);
    }

    protected override int GetHandlerLevel()
    {
        return 3;
    }
}
```

10:

a: Chain of Responsibility

b:

```
interface IGreetingHandler
{
	bool CanHandle(string nationality);
	void HandleGreeting(string name);
}

class EnglishGreetingHandler : IGreetingHandler
{
	bool CanHandle(string nationality){
		return nationality == ‘USA’;
	}
}
	void HandleGreeting(string name){
		Console.WriteLine($"Hello, {name}!");
	}
}
class SpanishGreetingHandler : IGreetingHandler
{
	bool CanHandle(string nationality){
		return nationality == ‘Spain’;
	}
}
	void HandleGreeting(string name){
		Console.WriteLine($"¡Hola, {name}!");
	}
}
class JapaneseGreetingHandler : IGreetingHandler
{
	bool CanHandle(string nationality){
		return nationality == ‘Japan’;
	}
}
	void HandleGreeting(string name){
		Console.WriteLine($"こんにちは, {name}!");
	}
}
```

```
class GreetingSystem{
	private List<IGreetingHandler> Handlers;

	public GreetingSystem()
	{
		this.Handlers = new List<IGreetingHandler>();
	}
	public void AddHandler(IGreetingHandler newHandler)
	{
		Handlers.Add(newHandler)
	}
	public void Greet(string nationality, string name){
		for( IGreetingHandler handler in Handlers)
		{
			if(handler.CanHandle(nationality))
			{
				handle.HandleGreeting( name );
				return;
			}
		}
		Console.WriteLine("Nationality not supported.");
	}
}
class Program
{
   static void Main()
   {
      GreetingSystem greetingSystem = new GreetingSystem();

      greetingSystem.RegisterGreetingHandler(new EnglishGreetingHandler());
      greetingSystem.RegisterGreetingHandler(new SpanishGreetingHandler());
      greetingSystem.RegisterGreetingHandler(new JapaneseGreetingHandler());

      greetingSystem.Greet("USA", "John");
      greetingSystem.Greet("Spain", "Juan");
      greetingSystem.Greet("Japan", "Yuki");
   }
}
```