MOQ ASSIGNMENT

Quesiton-1

You have an ICalculator interface.

```
public interface ICalculator
{
    int Add(int a, int b);
}
```

- Mock ICalculator.
- Setup Add(2, 3) to return 5.
- Verify the method was called exactly once.

Question-2

```
public interface ICustomerRepository
{
    Customer GetCustomerById(int id);
}

public class CustomerService
{
    private readonly ICustomerRepository _repo;

    public CustomerService(ICustomerRepository repo)
    {
        _repo = repo;
    }

    public string GetCustomerName(int id)
    {
        var customer = _repo.GetCustomerById(id);
        return customer?.Name ?? "Unknown";
    }
}
```

- Mock ICustomerRepository.
- Setup GetCustomerById(1) to return a Customer with name "John".
- Write a test that asserts "John" is returned from GetCustomerName(1).

Question-3

- Setup GetCustomerById(-1) to throw an ArgumentException.
- Assert that GetCustomerName(-1) throws or handles the exception gracefully.

Question-4

Mocking a Method with out or ref

```
public interface IParser
{
    bool TryParse(string input, out int number);
}
```

- Setup TryParse("123", out int) to return true and output 123.
- Write a test using Moq to mock this behavior and verify it.

Question-5

Verifying Call Count

```
public interface ILogger
{
    void Log(string message);
}
public class Processor
{
    private readonly ILogger _logger;
    public Processor(ILogger logger)
    {
        _logger = logger;
    }

    public void Process()
    {
        _logger.Log("Start");
        _logger.Log("Processing");
        _logger.Log("End");
    }
}
```

Write a test to verify that Log() was called **3 times**.

Question-6

```
public interface IConfig
{
    string Environment { get; set; }
}
```

- Mock IConfig.
- Set Environment = "Production".
- Verify that Environment property is used in a class that checks if(config.Environment == "Production").

Question-7 Callback in Mock

```
public interface INotifier
{
    void Notify(string message);
}
```

- Use Callback to capture the message passed to Notify().
- Assert the message content is correct.

Question-8

Mocking a Sequence of Returns

```
public interface IDataReader
{
    string ReadLine();
}
```

- Mock ReadLine() to return: "First", "Second", "Third" on consecutive calls.
- Test the method that reads 3 lines and collects them into a list.

Question-9

Setup Method with Parameters Using It.IsAny

```
public interface IDiscountService
{
    decimal ApplyDiscount(decimal amount);
}
```

- Mock ApplyDiscount(It.IsAny<decimal>()) to return 90% of the input.
- Test that ApplyDiscount(100) returns 90.

Question-10

- Create a mock with MockBehavior.Strict.
- Don’t set up one of the method calls.
- Verify the test fails when the unexpected method is called.