DEPI Technical Final Quiz

- **Which statement about the IDisposable interface is TRUE?**
  - It provides a mechanism for releasing unmanaged resources manually, often used in conjunction with the using statement.
  - It automatically releases all unmanaged resources when the object is garbage collected.
  - It is used to define classes that support asynchronous operations.
  - It is used only for managing memory.

- **Which statement about the lock statement in C# is TRUE?**
  - The lock statement is primarily used for asynchronous programming.
  - The lock statement creates a new thread to handle concurrent execution.
  - The lock statement can only be used with reference types, not value types.
  - **The lock statement is used to guarantee that a single thread can access a resource at any given time**.

- **Which feature of C# generics allows you to specify that a type parameter must implement an interface or inherit from a base class?**
  - Covariance
  - Polymorphism
  - **Type constraints**
  - Delegates

- **Which of the following statements about async and await is FALSE?**
  - **Methods marked with async automatically run on a different thread**.
  - await pauses the execution of a method until the awaited task is completed.
  - The async keyword can be applied to methods that return void.

- await can only be used inside methods marked with the async keyword.

- ## What can be achieved using Reflection in C#?
  - Directly access private fields without needing explicit permissions.
  - **Access and invoke methods or properties of an object at runtime**.
  - Create new value types at runtime.
  - Serialize an object into JSON format.

- ## Which of the following best describes how deferred execution works in LINQ?
  - LINQ queries are compiled at runtime to improve performance.
  - LINQ queries are executed when the results are enumerated.
  - **LINQ queries are executed immediately when defined**.
  - LINQ queries are executed in parallel by default.

- ## Which of the following is TRUE about model binding in ASP.NET MVC?
  - Model binding can only bind data from HTTP GET requests.
  - **Model binding automatically maps form data, route values, and query string parameters to controller action parameters.**
  - Model binding does not support custom complex types.
  - Model binding only works with primitive types such as int and string.

- ## In ASP.NET MVC, how does attribute routing differ from conventional routing?
  - Attribute routing supports only static routes, while conventional routing supports dynamic routes.
  - **Attribute routing allows routes to be defined directly on action methods, while conventional routing uses a central route table**.

- Conventional routing allows for more fine-grained control over route matching compared to attribute routing.
- Attribute routing is the only way to handle optional parameters in URLs.

- **Which of the following statements about action filters in ASP.NET MVC is FALSE?**
  - Action filters allow you to execute code before and after controller actions.
  - **Action filters only run on actions that return views**.
  - You can create custom action filters by inheriting from the ActionFilterAttribute class.
  - Action filters can be applied globally, to a controller, or to individual actions.

- **Output ?**
  - 20, 30
  - 20, 10
  - 30, 10
  - **30, 30**

```
public class Program
{
    public static void Main()
    {
        int x = 10;
        int y = 20;
        Console.WriteLine(Add(ref x, y));
        Console.WriteLine(x);
    }

    static int Add(ref int a, int b)
    {
        a += b;
        return a;
    }
}
```

- ## what will happend
  - The code won't compile.
  - The code will compile and print 0.
  - **A NullReferenceException will occur**.
  - The code will compile and print null.

```
public class Program
{
    public static void Main()
    {
        string? message = null;
        Console.WriteLine(message.Length);
    }
}
```

- ## Output

```
public class Program
{
    public static void Main()
    {
        var numbers = new List<int> { 1, 2, 3 };
        var result = numbers.Where(x => x > 1);

        numbers.Add(4);

        foreach (var item in result)
        {
            Console.WriteLine(item);
        }
    }
}
```

  - **2,3**

- ## what will happend

- **Derived Show"**
- "Base Exception"
- No output will be displayed.
- The code will not compile.

```
public class Base
{
    public virtual void Show() => throw new Exception("Base Exception");
}

public class Derived : Base
{
    public override void Show() => Console.WriteLine("Derived Show");
}

public class Program
{
    public static void Main()
    {
        Base obj = new Derived();
        try
        {
            obj.Show();
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```
