1. Select the correct statement among the given statements?
    1. C#.NET support partial implementation of interfaces.
    2. Properties could be declared inside an interface.
    3. Interface method calls are resolved at runtime using the actual object type.
    4. One interface can be implemented in another interface.
    5. Access specifiers that can be used for interface are public, protected and private.

- A. 1,2,4
- B. 1,2,3
- C. 2,3
- D. 1,4,5
- Answer: C

---

2. Choose the incorrect statement:

- A. To implement delegates necessary condition is class declaration.
- B. A single delegate can invoke more than one method.
- C. delegates could be shared.
- D. delegate is a value type.
- E. Delegates allow methods to be passed as parameters.
- Answer: D

---

3. What will be the correct way to implement the interface in the following C# code?

```
interface person {
    string name {
        get;
        set;
    }
}
```

- A.

```
class emp: person {
    private string str;
    public string name;
    {
        get { return str; }
        set { str = value; }
    }
}
```

- B.

```
class emp: implements person
{
    private string str;
    public string name
    {
        get { return str; }
        set { str = value; }
    }
}
```

- C.

```
class emp: person {
    private string str;
```

```
        public string person.name
        {
            get { return str; }
            set { str = value; }
        }
    }
```

- D: None of the mentioned
- Answer: D

---

4. A class FileLogger needs to implement a contract that includes:
    - A method: Log(string message)
    - A property: LogLevel { get; set; }
    - Support for multiple inheritance
    - No default implementation for members
    - Which of the following is the correct approach?

- A. Use an abstract class with virtual methods and override them in FileLogger.
- B. Use an interface with default method implementations and explicitly implement LogLevel.
- C. Use an interface without default implementations and let FileLogger provide all members.
- D. Use a static class with extension methods for Log().
- Answer: C

---

5. What will happen when this code is executed?

```
public class Test {
    sealed int secret = 42;
    public void Reveal() {
        Console.WriteLine(secret);
    }
}

public class Program {
    public static void Main() {
        Test test = new Test();
        test.Reveal();
    }
}
```

A. outputs 42 B. Compile error: "sealed" cannot be applied to fields C. 0 D. Runtime error: sealed fields require initialization in constructor

- Answer: B

---

6. What is the correct output?

```
using System;
interface IJump {
    void Jump();
}
abstract class Animal {
    public virtual void Jump() {
        Console.Write("Animal jumps ");
    }
    public abstract void Sound();
}
class Dog : Animal, IJump {
    public override void Jump() {
        Console.Write("Dog jumps ");
    }
    public override void Sound() {
        Console.Write("bark ");
    }
}
class Program {
    static void Main() {
        Animal myPet = new Dog();
        myPet.Jump();
        myPet.Sound();
    }
}
```

- A. "Animal jumps bark"
- B. "Dog jumps bark"
- C. Compile error
- D. "Animal jumps Dog jumps bark"
- Answer: B

---

7. Which statement about this property is true?

```
public int Id { get; init; }
```

- A. The property can be modified at any time by methods within the same class.
- B. After object initialization, attempting to change Id will succeed without errors.
- C. The init accessor restricts setting the property to object initialization only.
- D. This behaves identically to a property with a public set accessor.

- Answer: C

---

8. Which statement about this class is false?

```
public abstract class Processor {
    public abstract void Validate();
    public void Log(string message) {
        Console.WriteLine(message);
    }
    public virtual void Preprocess() {
        Console.WriteLine("Base preprocessing");
    }
}
```

- A. A derived concrete class must implement Validate() but can optionally override Preprocess()
- B. The Log() method can be hidden in a derived class using the new keyword
- C. The Preprocess() method must be overridden in any derived concrete class
- D. A derived abstract class can defer implementing Validate() to further child classes
- Answer: C

---

9. What are the limitation of generics in C#?
    1. Enums cannot have generic type parameters.
    2. Lightweight dynamic methods cannot be generic.
    3. Using generics with value types always avoids boxing entirely.
    4. Generic types cannot use arithmetic operators directly.
    5. Each generic type instantiation gets its own independent static fields.

- A. 1,3,5

- B. 1,2,4
- C. 2,4,5
- D. 1,2,3
- Answer: B

1. **Enums cannot have generic type parameters.**
   - Correct. Enum definitions in C# cannot include generic type parameters (e.g., `enum MyEnum<T> {...}` is invalid).
2. **Lightweight dynamic methods cannot be generic.**
   - Correct. Dynamic methods created via `System.Reflection.Emit.DynamicMethod` cannot be generic. This is a limitation of the reflection emit API.
3. **Using generics with value types always avoids boxing entirely.**
   - Incorrect. While generics *usually* avoid boxing for value types, boxing can still occur in scenarios like interface constraints (e.g., `where T : IComparable`), converting to `object`, or using `Enum`/`Delegate` types.
4. **Generic types cannot use arithmetic operators directly.**
   - Correct. Arithmetic operators (e.g., `+`, `-`, `*`) cannot be applied directly to generic type parameters without constraints (e.g., `T result = a + b;` fails). Workarounds require advanced constraints (e.g., `INumber<T>`) or dynamic methods.
5. **Each generic type instantiation gets its own independent static fields.**
   - Incorrect. This is **not a limitation** but a deliberate feature. Each closed generic type (e.g., `MyClass<int>`, `MyClass<string>`) has separate static fields, which is intentional behavior.

- **Correct Options**:
  - **1** (Enums cannot have generic type parameters)
  - **2** (Lightweight dynamic methods cannot be generic)
  - **4** (Generic types cannot use arithmetic operators directly)

---

10. Which keyword should be used when we need: "A field that cannot be modified after initialization"?

- A. Only const
- B. Only readonly
- C. Either const or readonly (depending on requirements)
- D. sealed
- Answer: C
