1. Primitive Type 에 관한 내용 중, 옳은 것을 모두 고르세요. 없으면 '없음' 으로 답하세요

1) char 타입은 int 타입과 변환이 불가능하다
2) 문자 '1' 은 정수타입 1 과 같다
3) 타입의 표현범위를 넘어서는 연산을 해도 *오류가 발생하지 않는다
4) 정수타입 대신 enum 타입을 사용하면 프로그램의 성능이 개선된다
5) 수식에서 산술연산자는 우선순위와 관계없이 왼쪽부터 계산한다

답 : 3

1) char <-> int 타입 캐스트 가능
2) 코드표에 따라 변환되며 문자 1은 정수 49
3) overflow 가 일어나지만, 오류가 발생하진 않는다
4) 열거형의 사용과 프로그램의 성능과는 관계 없다
5) 산술연산자의 우선순위에 따라 계산된다

2 - 1. 아래 코드의 실행을 예상하여 출력 결과를 적으세요

```
A a1 = new A(){ Number = 1 };
A a2 = new A(){ Number = 1 };

if( a1 == a2 )
        Console.WriteLine("a1 is a2");
else
        Console.WriteLine("a1 is not a2");
```

답 : a1 is not a2

> a1 과 a2 는 서로 다른 인스턴스이며, 비교연산자 == 는 참조가 같은지 비교하므로

2 - 2. 아래 코드에서 *오류가 발생하는 지점이 있다면 설명하고, 없다면 '없음' 으로 답하세요

```
A a3 = null;

if( a3.Number == 0 || a3 != null )
{
        Console.WriteLine($"Number of a3 is {a3.Number}");
}
```

답 : a3.Number == 0 지점에서 오류가 발생.
> a3 = new A(); 인스턴스를 할당하거나,
> a3 != null || a3.Number == 0 비교 순서를 바꾼다

3. 아래 코드에서 *오류가 발생하는 지점이 있다면 설명하고, 없다면 '없음' 으로 답하세요.

```
for (int i = 0; ; i++)
{
        Console.WriteLine(i);
}
```

답 : 오류가 발생하지 않는다. (무한루프이지만, 시험에서의 *오류 의 정의 상기)
> for 문의 initializer , condition, iterator 작성은 선택사항이다.

4. 객체 지향 프로그래밍에 관한 설명 중, 틀린것을 모두 고르세요. 없으면 '없음' 으로 답하세요

1) 자동차, 의자, 사물 등 시각적으로 구분 할 수 있는 것만 객체로 정의가 가능하다
2) 실재하는 대상을 객체로 다룰 때는 관점이 달라도 의미는 항상 유지된다
3) 절차지향 프로그래밍에 비해 성능이 좋기 때문에 선호된다
4) 객체를 개별적으로 수정하긴 어렵지만 재사용성이 용이하다
5) 추상 클래스는 인스턴스화 되지 않는다

답 1,2,3,4

1) 시각적인것 눈에 보이지 않는 기능이나 현상도 객체로 정의가 가능하다
2) 객체로 다룰 때, 관점이 달라지면 의미도 바뀔 수 있다.
3) 객체지향 프로그래밍은 수행 성능 개선을 위한 방법론은 아니다
4) 객체는 개별적으로 수정 가능하며, 재사용성도 용이하다
5) 추상 클래스는 인스턴스화 되지 않는다

5 - 1. 상속의 개념을 활용하여 개선하세요

```
class ABBase
{
    private int _number = 0;
    public int Number => _number;

    public void Add(ABBase t)
    {
        _number += t.Number;
    }
}

class A : ABBase
{
    public void Hellow() { }
}

class B : ABBase
{
    public void Byebye() { }
}
```

> 공통된 변수와 메소드를 부모클래스에 정의하고 활용하였는가

> C# 의 상속 문법을 이해하였는가

5 - 2. 코드블럭(2) 가 포함되도록 인터페이스를 활용하여 개선하세요

```
interface Adder
{
    public void Add(ABBase t);
}

class ABBase : Adder
{
    private int _number = 0;
    public int Number => _number;

    public void Add(ABBase t)
    {
        _number += t.Number;
    }
}
```

답1 > 부모 클래스에서 인터페이스를 구현한 경우

> 인터페이스의 정의 방법을 이해하였는가
> 인터페이스를 상속한 경우 구현의 필수사항을 이해하였는가

```
interface Adder
{
    public void Add(ABBase t);
}

class ABBase
{
    protected int _number = 0;
    public int Number => _number;
}

class A : ABBase,Adder
{
    public void Add(ABBase t)
    {
        _number += t.Number;
    }
    public void Hellow() { }
}

class B : ABBase,Adder
{
    public void Add(ABBase t)
    {
        _number += t.Number;
    }
    public void Byebye() { }
}
```

답2 > 자식 클래스에서 인터페이스를 각각 구현한 경우.

6 - 1. IHandshake 를 상속한 어떤 클래스 A 에 대해, 아래 코드에서 틀린 부분을 모두 고르세요. 없으면 '없음' 으로 답하세요. (틀린 부분이 있는경우, 그 이유도 서술하세요)

```
IHandshake shaker1 = new A();
shaker1.Hi();
A shaker2 = new A();
shaker2.Bye();
(shaker2 as IHandshake).Hi();
A shaker3 = new IHandshake();
shaker3.Hi();
IHandshake ishaker2 = shaker2;
ishaker2.Bye();
```

답 > A shaker3 = new IHandshake();
- 추상 클래스와 인터페이스는 인스턴스화 되지 않는다

6 - 2. HowAreYou() 메소드를 가진 클래스 A 와, FineThankAndYou() 메소드를 가진 클래스 B 가 모두 IHandshake 를 상속하도록 정의하세요

```
class A : IHandshake
{
    public void Hi() { }
    public void Bye() { }
    public void HowAreYou() { }
}
```

```
class B : IHandshake
{
    public void Hi() { }
    public void Bye() { }
    public void FineThankAndYou() { }
}
```

6 - 3. 6-2 에서 정의한 클래스가 아래 코드블럭에서 동작하도록 빈 칸 **[-n-]** 을 채우세요

```
var handshakers = new List<[-1-]>();
handshakers.Add([-2-]);
handshakers.Add([-3-]);

for(int i = 0; i < handshakers.Count ; i++)
        handshakers[i].Hi();

([-4-]).HowAreYou();

var handshaker2 = handshake[1];
if ( handshaker2 [-5-] )
        ( handshaker2 [-6-] ).FineThankAndYou();
```

```
var handshakers = new List<IHandshake>(); // [1]
handshakers.Add(new A());  // [2]
handshakers.Add(new B());  // [3]

for (int i = 0; i < handshakers.Count; i++)
    handshakers[i].Hi();

(handshakers[0] as A).HowAreYou(); // [4]
var handshaker2 = handshakers[1];
if (handshaker2 is B) // [5]
    (handshaker2 as B).FineThankAndYou(); //[6]
```

> [4] , [6] 번은 괄호() 로 캐스트 한 것도 정답 인정

```
((A)handshakers[0]).HowAreYou(); // [4]
var handshaker2 = handshakers[1];
if (handshaker2 is B) // [5]
    ((B)handshaker2).FineThankAndYou(); //[6]
```