Odpovědi pište na zvláštní odpovědní list s vaším jménem a fotografií. Pokud budete odevzdávat více než jeden list s řešením, tak se na 2. a další listy nezapomeňte podepsat. Do zápatí všech listů vždy napište i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

Společná část pro otázky označené A

Předpokládejte následující program:

```
var x1 = new X1();
if (x1.Value > 7 && x1.Value < 25 && x1.Value != 8)
{
  Console.WriteLine("OK1");
}
Console.WriteLine(" + ");
if (x1.Value is > 7 and < 25 and not 8) {
  Console.WriteLine("OK2");
}

class X1 {
  public int Value {
    get {
      Console.WriteLine("X");
      return 10;
} } }
```

Otázka č. 1 (A)

Co uvedený program vypíše? Detailně vysvětlete proč.

Otázka č. 2 (A)

Detailně vysvětlete, jaká je velikost proměnné `x1` a jakou má v paměti strukturu. Dále detailně vysvětlete, jaká je velikost instance `X1` a jak instance `X1` souvisí se samotnou proměnnou `x1`.

Společná část pro otázky označené B

V příloze 1 najdete kompletní obsah zdrojového souboru `Program.cs` z našeho projektu `FactorialBenchmark.csproj` se sadou čtyř benchmarků.

Otázka č. 3 (B)

Uvedené benchmarky jsme spustili na počítači s následující konfigurací: „`Windows 11, Intel Core 7 150U 1.80GHz, 1 CPU, 12 logical and 10 physical cores .NET SDK 10.0.101`“ a vyšly nám tyto výsledky:

Method	Mean	Error	StdDev	Median
Alpha0	0.6500 ns	0.0375 ns	0.0605 ns	0.6208 ns
Bravo0	0.0033 ns	0.0042 ns	0.0039 ns	0.0011 ns
Alpha2	1.1396 ns	0.0267 ns	0.0223 ns	1.1346 ns
Bravo2	1.1796 ns	0.0467 ns	0.0459 ns	1.1767 ns

Detailně vysvětlete, z jakého důvodu vychází benchmark `Bravo0` rychlejší než `Alpha0` (proč jejich výsledky nejsou víceméně shodné). Dále vysvětlete, proč je navíc mezi těmito dvěma výsledky tak velký rozdíl. Oba benchmarky počítají faktoriál pro stejnou hodnotu, a navíc se zdá, že oba způsoby výpočtu jsou ekvivalentní, tak se zdá zvláštní, že výsledky měření mají tak rozdílné.

Otázka č. 4 (B)

Předpokládejte, že si **do prázdného** adresáře zkopírujeme **pouze** výsledný soubor `FactorialBenchmark.dll`, a ten pak otevřeme v nástroji ILSpy. Napište C# kód metody `CalcFactorialBravoInternal` (včetně její celé hlavičky, a celého těla), jak ho nejspíše z CIL kódu dekompiluje ILSpy.

Otázka č. 5 (B)

Předpokládejte, že si v nástroji ILSpy otevřeme výsledný soubor `FactorialBenchmark.dll`. Napište kompletní C# kód, jak ho nejspíše z CIL kódu dekompiluje ILSpy, pro třídu nebo strukturu (včetně všech metod, fieldů, atd.), ve které skončí kód odpovídající následujícímu řádku původního zdrojového souboru:

```
BenchmarkRunner.Run<FactorialBenchmarks>();
```

Otázka č. 6 (B)

S využitím testovacího frameworku xUnit, MSTest, nebo nUnit napište sadu alespoň 3 vhodných relevantních unit testů testujících rozdílné aspekty chování metody `CalcFactorialAlpha`.

Společná část pro otázky označené C

Předpokládejte následující program:

```
C7? c7 = new E7();
Console.WriteLine(c7.f());

I7 i7 = new D7();
Console.WriteLine(i7.f());

interface I7 {
    public char f();
}
class A7 : I7 {
    public virtual char f() => 'A';
}
class B7 : A7 {
    public override char f() => 'B';
}
class C7 : B7 {
    public virtual char f() =>
        (char) (base.f() + 5);
}
class D7 : C7 {
    public override char f() => 'D';
}
class E7 : D7 {
    public override char f() =>
        (char) (base.f() + 10);
}
```

Otázka č. 7 (C)

Co uvedený program vypíše na 1. řádku výstupu? Detailně vysvětlete proč – jako součást vysvětlení nakreslete VMT tříd `A7`, `B7`, `C7`, `D7`, `E7`.

Otázka č. 8 (C)

Co uvedený program vypíše na 2. řádku výstupu? Detailně vysvětlete proč – jako součást vysvětlení nakreslete interfacovou tabulku metod pro interface `I7` u tříd `A7`, `B7`, `C7`, `D7`, `E7`.

Otázka č. 9

Následující program se bez chyb přeloží. Napište, co program po spuštění vypíše na standardní výstup a detailně vysvětlete proč.

```
class Prg9 {
   static void Main(string[] args) {
      try {
         try {
            throw new ArgumentException();
         } finally {
            Console.WriteLine("A");
            try {
               try {
                  throw new
                  IndexOutOfRangeException();
               } finally {
                  Console.WriteLine("B");
                  try {
                     try {
                        throw new
                        NotSupportedException();
                     } finally {
                        Console.WriteLine("C");
                     }
                  } catch (Exception ex) {
                     Console.WriteLine(
                        $"{ex.GetType()}"
                     );
                     throw;
                  }
                  Console.WriteLine("D");
               }
            } catch (Exception ex) {
               Console.WriteLine(
                  $"{ex.GetType()}"
               );
            }
            Console.WriteLine("E");
         }
      } catch (Exception ex) {
         Console.WriteLine($"{ex.GetType()}");
      }
   }
}
```

Otázka č. 10

Předpokládejte níže uvedenou třídu `Image` pro reprezentaci bitmapových obrázků, a její potomky pro načítání a ukládání obrázků z formátů JPEG a BMP. Ve třídě `Prg10` naleznete základní příklad použití daných tříd. Do budoucna bychom chtěli přidávat podporu pro další obrazové formáty, další algoritmy zpracování obrázků, a samozřejmě chceme mít možnost načítat obrázky v libovolném podporovaném formátu, a upravený obrázek uložit v libovolném formátu dle volby uživatele.

Vysvětlete, proč tento objektový návrh není vhodný, a **upravte** ho na vhodnější (včetně adekvátní úpravy metody `Main`). Vysvětlete, v čem je váš objektový návrh lepší. Metoda `UpdateImageForProjector` by měla **zůstat beze změn**.

```
abstract class Image {
  protected Color[,]? bitmap = null;

  public void ChangeBrightness(float amount) {
    // Implementace změny světlosti
  }

  public void ChangeConstrast(float amount) {
    // Implementace změny kontrastu
  }

  public abstract void Load(string fileName);
  public abstract void Save();
}

class JpegImage : Image {
  public override void Load(string fileName) {
    // Načítání JPEG obrázku do pole bitmap
  }
  public override void Save() {
    // Ukládání JPEG obrázku z pole bitmap
    // do souboru zapamatovaného jména
  }
}

class BmpImage : Image {
  public override void Load(string fileName) {
    // Načítání BMP obrázku do pole bitmap
  }
  public override void Save() {
    // Ukládání BMP obrázku z pole bitmap
    // do souboru zapamatovaného jména
  }
}

class Prg10 {
  public static void Main() {
    var image = new JpegImage();
    image.Load("blackboard.jpg");
    UpdateImageForProjector(image);
    image.Save();
  }

  public static void UpdateImageForProjector(
    Image image
  ) {
    image.ChangeBrightness(1.75f);
    image.ChangeConstrast(1.6f);
  }
}
```

```
1  using BenchmarkDotNet.Attributes;
2  using BenchmarkDotNet.Running;
3
4  BenchmarkRunner.Run<FactorialBenchmarks>();
5
6  public class FactorialBenchmarks {
7      // Variant Alpha:
8      public static int CalcFactorialAlpha(int sourceN) {
9          if (sourceN == 0) return 1;
10         if (sourceN == 1) return 1;
11         if (sourceN < 0) throw new ArgumentOutOfRangeException(nameof(sourceN), sourceN, "Should be
               non-negative.");
12
13         try {
14             int resultOfNFactorial = 1;
15             for (int i = 1; i <= sourceN; i++) {
16                 checked { resultOfNFactorial *= i; }
17             }
18             return resultOfNFactorial;
19         } catch (OverflowException) {
20             throw new ArgumentOutOfRangeException(nameof(sourceN), sourceN, "Too big for Int32
                   result.");
21         }
22     }
23
24     // Variant Bravo:
25     public static int CalcFactorialBravo(int sourceN) {
26         if (sourceN == 0) return 1;
27         if (sourceN == 1) return 1;
28
29         return CalcFactorialBravoInternal(sourceN);
30     }
31
32     private static int CalcFactorialBravoInternal(int sourceN) {
33         if (sourceN < 0) throw new ArgumentOutOfRangeException(nameof(sourceN), sourceN, "Should be
               non-negative.");
34
35         try {
36             int resultOfNFactorial = 1;
37             for (int i = 1; i <= sourceN; i++) {
38                 checked { resultOfNFactorial *= i; }
39             }
40             return resultOfNFactorial;
41         } catch (OverflowException) {
42             throw new ArgumentOutOfRangeException(nameof(sourceN), sourceN, "Too big for Int32
                   result.");
43         }
44     }
45
46     // Actual benchmarks:
47     [Benchmark]
48     public int Alpha0() => CalcFactorialAlpha(0);
49
50     [Benchmark]
51     public int Bravo0() => CalcFactorialBravo(0);
52
53     [Benchmark]
54     public int Alpha2() => CalcFactorialAlpha(2);
55
56     [Benchmark]
57     public int Bravo2() => CalcFactorialBravo(2);
58 }
```