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).

Otázka č. 1

Předpokládejte, že chceme naimplementovat následující třídu `MySemaphore`, jejíž základní API odpovídá standardnímu chování třídy `System.Threading.SemaphoreSlim`:

```
public class MySemaphore {
   public MySemaphore(int initialCount) {
   }

   /// <returns>The previous count of
   /// the Semaphore.</returns>
   public int Release() {
   }

   public void Wait() {
   }
}
```

Napište implementaci takové třídy `MySemaphore` pouze s využitím třídy `Monitor` a jejích metod (možno použít i nepřímo konstruktem jazyka C#) bez použití `SemaphoreSlim` i bez použití `WaitHandle` a jejích libovolných potomků.
[1,5 bodu]

Otázka č. 2

Předpokládejme následující část kódu, který se bez chyb přeloží (navíc předpokládejte standardní .NET memory model):

```
using System;

public class X {
    public volatile int a;
    public long b;
    public volatile int c;

    public void T1() {
        a = 1;
        b = 2;
        c = 3;
    }
    public void T2() {
        b = -1;
        Console.Write(b);
    }
}
```

Na společné instanci třídy `X` spustíme současně 2 vlákna – 1. vlákno s metodou `T1` , 2. vlákno s metodou `T2`.

(A) Je z pohledu vlákna `T2` pravda, že zápis do `b` v `T1` vždy proběhne před zápisem do `c`? Vysvětlete proč!
[1 bod]

(B) Jaké hodnoty může `T2` vypsat na standardní výstup? Vysvětlete proč.
[1 bod]

Otázka č. 3

Předpokládejte následující program uložený ve zdrojovém souboru `Program.cs`:

```
class State {
   public int count;
}

public class Prg3 {
   static IEnumerable<State> GetState() {
      yield return new State { count = 0 };
   }

   public static int Foo(int a, int b) {
      while (a < b) {
         if (a == b - 3) {
            Console.WriteLine(a);
         }
         a++;
      }
      return a;
   }

   public static void Main() {
      var result =
        (from s in GetState()
            from l in File.ReadLines("Program.cs")
            select new { s, l }
        ).SkipWhile(x =>
            !x.l.Contains("Foo") ||
            !x.l.EndsWith("{")
        ).TakeWhile(x => {
            if (x.l.Contains("{")) x.s.count++;
            var c = x.s.count;
            if (x.l.Contains("}")) x.s.count--;
            return c > 0;
        }).Select(x => x.l);

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

(A) Napište, co program vypíše na standardní výstup. Je to nejvhodnější zápis uvedeného algoritmu pomocí LINQ to Objects?
[1 bod]

(B) Detailně vysvětlete, jakým způsobem se bude tento program chovat, jakým způsobem bude probíhat vyhodnocení LINQ dotazu (nakreslete obrázek), a pro jednotlivé relevantní části kódu vysvětlete, ve kterých vláknech se budou provádět. Metoda `File.ReadLines` má následující deklaraci:

```
IEnumerable<string> File.ReadLines(string path)
```

[1 bod]

Otázka č. 4

Metadata assembly `Library.dll` se v nástroji `ildasm` zobrazí následujícím způsobem:

Dále lze zjistit, že implementace metody `A.m` je v CIL (MSIL) assembleru následující:

```
.method public hidebysig static int32 m(int32 x,
                                         int32 y)
cil managed
{
  // Code size       18 (0x12)
  .maxstack  8
  IL_0000:  br.s       IL_000c
  IL_0002:  ldarg.0
  IL_0003:  ldc.i4.2
  IL_0004:  add
  IL_0005:  starg.s    x
  IL_0007:  ldarg.1
  IL_0008:  ldc.i4.4
  IL_0009:  sub
  IL_000a:  starg.s    y
  IL_000c:  ldarg.0
  IL_000d:  ldarg.1
  IL_000e:  bne.un.s   IL_0002
  IL_0010:  ldarg.0
  IL_0011:  ret
}
```

(A) Zapište v C# deklaraci a tělo takové metody `A.m` (která se bude chovat stejně jako výše uvedený kód).
[1 bod]

(B) Bylo by možné v C# napsat program, který by v uvedené assembly `Library.dll` změnil tělo metody `A.m` tak, že by vždy na standardní výstup vypsala jen hodnotu proměnné `v`, a vrátila hodnotu `0`? Pokud ano, jaký nástroj/API byste k tomu nejlépe použili?
[0,5 bodu]

Otázka č. 5

Předpokládejte následující třídu:

```
class X {
   private static Task f(int a) { ... }
   private static Task<int> g(int a, int b) { ... }
   private static int tx(int x) { ... }

   public static async Task<int> m(int x, int y) {
      f(x);
      int r = await g(x + 1, tx(y));
      return r * 2;
}  }
```

Přepište třídu `X` do ekvivalentního kódu bez použití async/await patternu (tj. využijte pouze task continuations).
[1,5 bodu]

Otázka č. 6

Opravte/doplňte/doprogramujte nejvhodnějším způsobem následující kód tak, aby třída `X` byla serializovatelná s použitím standardní binární serializace. Předpokládejte, že deserializace probíhá v aplikaci běžící na webovém serveru, a že zaserializované instance tříd `X` submitují uživatelé pomocí webového rozhraní stejné aplikace.

```
enum Color {
    Red = 0, Green = 1, Blue = 2,
    First = Red, Last = Blue
};

class X {
    private Color c1 = Color.Red;
    private Color c2 = Color.Red;

    private void ThrowIfColorIsInvalid<TEx>(
        Color c
    ) where TEx : Exception, new()
    {
        if (c < Color.First || c > Color.Last) {
            throw new TEx();
        }
    }

    public Color Color1 {
        get { return c1; }
        set {
            ThrowIfColorIsInvalid<
            ArgumentOutOfRangeException>(value);
            c1 = value;
    }   }

    public Color Color2 {
        get { return c2; }
        set {
            ThrowIfColorIsInvalid<
            ArgumentOutOfRangeException>(value);
            C2 = value;
    }   }
}
```

[1,5 bodu]