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` bez použití `SemaphoreSlim` i bez použití `WaitHandle` a jejích libovolných potomků.

[1,5 bodu]

Otázka č. 2

Předpokládejte následující program:

```
public class Prg2 {
   public static void Main() {
      int[][] data = {
         new [] {1, 5, 3},
         new [] {-2},
         new int[0],
         new [] {-1, 3, 2, 3}
      };

      var x = from a in data
              let max = a.DefaultIfEmpty(-1).Max(
                 i => i * 10
              ) where max >= a.Length select max;

      foreach (var m in x) {
         Console.WriteLine(m);
      }
} }
```

(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.

[1 bod]

Otázka č. 3

Předpokládejte následující program:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.IO;

class Prg3 {
   static void Main(string[] args) {
      var s = new Socket(
         AddressFamily.InterNetwork,
         SocketType.Stream,
         ProtocolType.Tcp
      );
      s.Bind(new IPEndPoint(
         IPAddress.Loopback, 1234
      ));
      s.Listen(10);

      var c = s.Accept();
      s.Close();

      using (var sr = new StreamReader(
         new NetworkStream(
            c, FileAccess.Read,
            ownsSocket: false
      ))) {
         using (var sw = new StreamWriter(
            new NetworkStream(
               c, FileAccess.Write,
               ownsSocket: false
         ))) {
            int ch;
            while ((ch = sr.Read()) != -1) {
               sw.Write(char.ToUpper((char) ch));
            }
            sw.WriteLine();
            sw.WriteLine("Server down.");
         }
      }

      c.Close();
} }
```

(A) Uvedený program spustíme na nějakém počítači. Dále předpokládejte, že na stejném počítači současně na příkazové řádce spustíme standardní telnet klient pomocí:

```
telnet 127.0.0.1 1234
```

Na standardním vstupu tohoto telnetového klienta pak zadáme následující písmena:

```
a h o j
```

(tedy na klávesnici stiskneme uvedené 4 klávesy). Popište, co se objeví na standardním výstupu telnetového klienta a vysvětlete proč!

[1 bod]

(B) Jakým způsobem je třeba uvedený C# program upravit, aby fungoval lépe (správněji). Vysvětlete.

[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       16 (0x10)
  .maxstack  2
  .locals init ([0] int32 z)
  IL_0000:  ldarg.0
  IL_0001:  ldarg.1
  IL_0002:  bge.s      IL_000a
  IL_0004:  ldarg.0
  IL_0005:  ldc.i4.2
  IL_0006:  add
  IL_0007:  stloc.0
  IL_0008:  br.s       IL_000e
  IL_000a:  ldarg.1
  IL_000b:  ldc.i4.1
  IL_000c:  add
  IL_000d:  stloc.0
  IL_000e:  ldloc.0
  IL_000f:  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 {
   public static async Task<int> m(int a, int b) {
      var sr = new StreamReader("x.txt");

         string line;
         int sum = 0;
         while (
            (line = await sr.ReadLineAsync())
            != null
         ) {
            var length = line.Length;
            if (line.StartsWith(">>>")) {
               line = await sr.ReadLineAsync();
               sum += Math.Max(
                  length, line.Length
               );
            }
         }

         sr.Dispose();
         return sum;
} }
```

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

Otázka č. 6

Předpokládejte, že implementujeme jednoduchý ORM framework. Vaším úkolem je napsat implementaci třídy `ObjectProxy` a její metody `SetFields`, která má sloužit pro naplnění hodnot datových položek předpřipravené instance libovolného typu `T` (neznámého za překladu třídy `ObjectProxy`). Prvním argumentem metody `SetFields` je právě taková instance, druhým argumentem je seznam dvojic: jméno datové položky + hodnota, která má být do dané položky uložena. Ve své implementaci vyjděte z níže uvedené kostry a neošetřujte žádné chybové stavy:

```
public class ObjectProxy<T> {
   public void SetFields(
      T instance,
      Dictionary<string, object> fieldValues
   ) {

   }
}
```

[1,5 bodu]