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 následující třídu:

```
public class SortedList<T> : IEnumerable<T> {
   private IComparer<T> comparer;
   private Node root = null;

   public SortedList() {
      comparer = Comparer<T>.Default;
   }

   public SortedList(IComparer<T> customComparer) {
      comparer = customComparer;
   }

   public void Add(T value) {
      var node = new Node { Value = value };
      if (root == null) {
         root = node;
      } else {
         var target = root;
         while (true) {
            if (comparer.Compare(
                value, target.Value
               ) < 0) {
               if (target.Left == null) {
                  target.Left = node;
                  break;
               } else {
                  target = target.Left;
               }
            } else {
               if (target.Right == null) {
                  target.Right = node;
                  break;
               } else {
                  target = target.Right;
               }
   } } } }

   public IEnumerator<T> GetEnumerator() {
      Stack<Node> stack = new Stack<Node>();

      Node current = root;
      while (current != null || stack.Count != 0) {
         while (current != null) {
            stack.Push(current);
            current = current.Left;
         }
         while (current == null
                && stack.Count != 0) {
            current = stack.Pop();
            yield return current.Value;
            current = current.Right;
         };
   } }

   IEnumerator IEnumerable.GetEnumerator() {
      return GetEnumerator();
   }

   private class Node {
      public T Value;
      public Node Left = null;
      public Node Right = null;
   }
}
```

Uvedený kód obsahuje naši iniciální implementaci setříděného seznamu pomocí binárního vyhledávacího stromu. Očekáváme, že v budoucnu budeme třídu dále vylepšovat minimálně o možnost vyhledávání prvků, a jejich odebírání. Dále v dalších verzích plánujeme vylepšit samotnou datovou strukturu o nějakou variantu vyvažování binárního stromu (nejspíše přejdeme na AVL nebo červeno-černý strom). Přepište třídu tak, aby byla serializovatelná a deserializovatelná pomocí standardní .NET binární serialiace (vyberte nejvhodnější způsob implementace, a uveďte pouze rozdíly oproti verzi ze zadání). Vysvětlete, proč je vámi zvolená implementace ta nejvhodnější.

[1,5 bodu]

Otázka č. 2

Předpokládejte následující třídu:

```
public class X : INotifyPropertyChanged {
   public event
      PropertyChangedEventHandler PropertyChanged;

   private string name;
   public string Name {
      get { return name; }
      set {
         name = value;
         if (PropertyChanged != null) {
            PropertyChanged(this,
               new PropertyChangedEventArgs(
                  "Name"
               ));
   } } }
   private double height;
   public double Height {
      get { return height; }
      set {
         height = value;
         if (PropertyChanged != null) {
            PropertyChanged(this,
               new PropertyChangedEventArgs(
                  "Height"
               ));
} } } }
```

Rozhodněte, zda je třída X thread-safe (bez ohledu na kontext použití). Vysvětlete, proč ano, resp. proč ne.

[1,5 bodu]

Otázka č. 3

Předpokládejte, že chceme vyrobit program, který bude uživateli zobrazovat filtrovaný seznam zákazníků nějaké české firmy. V aplikaci bude k dispozici `TextBox`, kde bude moci uživatel napsat začátek příjmení nějakého zákazníka. Po tomto zadání aplikace zobrazí jen ty zákazníky, jejichž příjmení začíná zadaným textem. Důležitý požadavek je, že vyhledávání v seznamu zákazníků musí podporovat zadání části příjmení s i bez diakritiky a s libovolnou velikostí písmen. Tedy zákazníka „Jiráček" musí aplikace vypsat např. po zadání „jirá", „JirÁ", „jirac", i „Jirácek". Vysvětlete, jak byste koncepčně takové vyhledávání implementovali.

[1,5 bodu]

Otázka č. 4

Předpokládejte následující program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.IO;

public class Prg4 {
   public static void Main() {
      IEnumerable<string> lines =
         File.ReadLines("in.txt");

      var a = (
         from l in lines
            where l.StartsWith("ERROR")
         from c in l.Split(',') select c
      ).AsParallel().Max(c => c.Length);

      Console.WriteLine(a);
   }
}
```

a následující testovací vstupní soubor `in.txt`:

```
WARNING,one,one
ERROR,two,three
WARNING,verylongword,andevenlongerword
ERROR,medium
ERROR,1,2,3,4,5
```

a) Rozhodněte, jaký bude typ proměnné a. Dále napište, co pro uvedená data program vypíše na standardní výstup. Bude výsledek programu na uvedených datech vždy stejný?
[0,5 bodu]

---

b) Detailně vysvětlete, jakým způsobem se bude tento program chovat, jakým způsobem bude probíhat vyhodnocení LINQ dorazu (nakreslete obrázek), a pro jednotlivé relevantní části kódu vysvětlete, ve kterých vláknech se budou provádět.
[1 bod]

---

c) Pokud trváme na maximálně paralelním provádění jednotlivých částí uvedeného LINQ dotazu, je uvedený LINQ dotaz ten nejvhodnější? Pokud ano, tak vysvětlete proč. Pokud ne, tak napište lepší alternativu.
[1 bod]

Otázka č. 5

Předpokládejte, že implementujeme základ webového serveru, který má mít maximální propustnost. Napište s využitím tříd `Socket`, nebo `TcpListener` a `TcpClient`, hlavní smyčku takové aplikace zařizující přijímání nových připojení od klientů. Předpokládejte, že pro samotnou komunikaci s konkrétním klientem již máte připravenou vhodnou metodu, která typicky zpracuje jeden požadavek a uzavře spojení (zpracování typického požadavku vyžaduje načtení souboru z disku, jeho případné zpracování na straně serveru, a odeslání upraveného obsahu klientovi). Napište i hlavičku takové vhodné metody, a popište její fungování.
[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]