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 datovou strukturu z RPG hry, kterou programujeme – rozhraní `IEntity` reprezentuje herní postavy; třída `Item`, a třídy od ní odděděné, reprezentují objekty, které může mít postava u sebe, a které mohou ovlivňovat její vlastnosti:

```
public interface IEntity {
    int FireResistance { get; set; }
    ...
}
public class Item {
    public virtual void ApplyEffectsOn(IEntity e){}
}

public class Gem : Item {
    public double Weight { get; set; }
}

public class FireGem : Gem {
    public int Power { get; set; }
    public override void ApplyEffectsOn(IEntity e)
    { e.FireResistance *= Power; }
}

public class Coin : Item {
    public int Value { get; set; }
}

public abstract class CompositeItem : Item {
    protected List<Item> items = new List<Item>();
    public IEnumerable<Item> Content => items;
    public void AddItem(Item item) { ... }
}

public class MoneyBag : CompositeItem {
    public int TotalAmount { get {
        int total = 0;
        foreach (Coin c in items) total += c.Value;
        return total;
}   }   }

public class ItemLoader {
    public List<Item> LoadFrom(TextReader r) {
       ...
}   }
```

(A) Třída `ItemLoader` slouží k načtení seznamu herních objektů z následujícího textového formátu – všimněte si, že textová reprezentace může být velmi specifická pro konkrétní druh objektu:

```
Gem: Weight=average
Coin: Value=2
Gem: Weight=tiny
FireGem: Weight=tiny, Power=2
MoneyBag:
    Coin: Value=1
    Coin: Value=5
```

Důkladně vysvětlete (a uveďte hrubou kostru kódu), jak byste navrhli třídu `ItemLoader` tak, abychom co nejsnadněji a nejefektivněji rozšiřovali čtecí algoritmus třídy `ItemLoader` o nové druhy objektů – těch budeme v aplikaci přidávat velké množství.

[1 bod]

(B) Bylo by možné přepsat třídy `CompositeItem` a `MoneyBag` tak, aby se nám v programu nemohlo stát, že bychom chybně do měšce (`MoneyBag`) vložili jinou položku než mince? Pokud ano, tak takové implementace napište a vysvětlete. Pokud ne, tak detailně vysvětlete proč.

[1 bod]

(C) Třída `CompositeItem` je označena jako `abstract`, přestože v sobě nemá žádné abstraktní metody. Je takový zápis správný? Pokud ano, tak vysvětlete, proč zde byl asi použit. Pokud ne, tak detailně vysvětlete proč.

[1 bod]

Otázka č. 2

Všimli jsme si, že pokud zapomeneme na instanci třídy `FileStream` zavolat metodu `Dispose`, tak přesto běžně dojde na konci programu k samočinnému zapsání dat, která zůstala ve vnitřním bufferu `FileStream`u, do souboru. Bohužel instance tříd `StreamWriter`, které `FileStream` využívají, se tak již nechovají – tj. pokud zapomeneme na `StreamWriter`u zavolat metodu `Dispose`, tak o explicitně nepropsaná data přijdeme. Vysvětlete, jaký je mezi těmito dvěma třídami rozdíl, a proč je každá chová odlišně. Bylo by možné `StreamWriter` „spravit“, aby se choval podobně jako `FileStream`? Vysvětlete proč.

[1 bod]

Otázka č. 3

Co vypíše následující program na standardní výstup? Detailně vysvětlete proč.

```
public interface I {
    void m(object o);
}

class A {
    public void m(object o) {
        Console.WriteLine("3.1");
}   }

class B : A {
    public void m(object o) {
        Console.WriteLine("3.2");
}   }

class C : B, I {
    public void m(System.String s) {
        Console.WriteLine("3.3");
}   }

class Prg3 {
    public static void Main() {
        I i = new C();
        i.m("Hello");
}   }
```

[1 bod]

Otázka č. 4

Předpokládejte, že verze 1 knihovny `X.dll` byla přeložena z následujícího zdrojového kódu:

```
public class X {
    public static void m(double x) {
        Console.Write("1");
}   }
```

Verze 2 knihovny `X.dll` byla přeložena z kódu:

```
public class X {
    public static void m(int x) {
        Console.Write("2");
}   }
```

Aplikace `A.exe` byla přeložena vůči verzi 2 `X.dll` z kódu:

```
class Prg4 {
    static void f() { X.m(1); }
    public static void Main() { f(); }
}
```

Rozhodněte a vysvětlete, jaké bude chování aplikace `A.exe`, pokud se ji pokusíme spustit, a v jejím adresáři bude k dispozici jen `X.dll` verze 1.

[1 bod]

Otázka č. 5

Předpokládejte následující program:

```
using C = System.Console;
class Prg5 {
    static void Main() {
        try {
            try {
                throw new DivideByZeroException();
            } finally {
                try {
                    throw
                    new ArgumentNullException();
                } catch (Exception ex1) {
                    C.WriteLine(ex1.GetType());
                }
            }
        } catch (Exception ex2) {
            C.WriteLine(ex2.GetType());
        }
}   }
```

Detailně vysvětlete, co a proč vypíše po svém spuštění.

[1 bod]

Otázka č. 6

Vysvětlete, jaké jsou v C# hlavní rozdíly mezi strukturami (`struct`) a třídami (`class`). V jaké situaci byste pro vaši datovou strukturu zvolili koncept struktury místo konceptu třídy? Detailně vysvětlete proč.

[1 bod]

Otázka č. 7

Co se stane po spuštění následujícího programu? Vypíše něco na standardní výstup (pokud ano, tak co)? Detailně vysvětlete proč.

```
class Prg7 {
    static int f(int a, int b) {
        return a + b;
    }

    public static void Main() {
        int x = 0x7FFFFFFF;
        int y = x;
        checked {
            x = f(x, 1);
            x -= 1;
            y += x;
        }
        System.Console.Write(y);
    }
}
```

[1 bod]

Otázka č. 8

Co vypíše následující program na standardní výstup? Detailně vysvětlete proč.

```
abstract class A { public abstract void m(); }

class B : A { public override void m() {
        System.Console.Write("8.B");
}   }

class C : B { public virtual void m() {
        System.Console.Write("8.C");
}   }

class D : C { public override void m() {
        System.Console.Write("8.D");
}   }

static class Prg8 {
    public static void Main() {
        A a = new D();
        ((B) ((C) a)).m();
}   }
```

[1 bod]