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 v rámci velkého projektu si potřebujeme udržovat jeden společný sdílený stav mezi různými instancemi většího množství tříd. Tento společný stav nám reprezentuje následující třída `GlobalState`:

```
public class GlobalState {
    private static int state = 0;
    public static int State => state;
    public static void AddToState(int a, int b) {
        state = MagicOMatic(a, b, state);
    }
    private static int MagicOMatic(
        int a, int b, int state
    ) { ... }
}
```

Následující třída `A` je ukázkou jedné z mnoha tříd našeho projektu, které budou sdílený stav využívat:

```
public class A {
    public int Value { get; private set; }
    public void Method(int x) {
        if (Value > 10) {
            GlobalState.AddToState(x, 0);
        }
        Value = x;
}   }
```

(A) Bylo by možné v C# místo `GlobalState` zapsat třídu s ekvivalentními vlastnostmi, ale bez použití/s minimem *statických* datových položek a metod? Pokud ne, vysvětlete proč. Pokud ano, tak příklad takové implementace napište (zároveň upravte kód třídy `A`, aby bylo zřejmé, jak budete novou implementaci `GlobalState` používat).
[1 bod]

(B) Pokud jste v bodě (A) odpověděli *ne*, tak napište příklad kódu, který by měl k řešení se statickými prvky nejblíže, ale přesto měl nějaké zásadní problémy – ty popište a vysvětlete. Pokud jste v bodě (A) odpověděli *ano*, tak vysvětlete, jaké všechny výhody by vaše řešení z bodu (A) mělo oproti řešení původnímu.
[1 bod]

Otázka č. 2

Předpokládejte, že programujeme informační systém pro mezinárodní řetězec mateřských školek, kde v rámci jeho implementace budeme pracovat s desítkami tisíc instancí třídy, která reprezentuje jedno dítě. Vysvětlete, zda je v dané situaci a pro níže uvedenou třídu implementace metody `GetHashCode` vhodná.

```
public class Child {
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Age { get; set; }
    public override int GetHashCode() {
        return Age;
}   }
```

[1 bod]

Otázka č. 3

Předpokládejte následující dvě různé implementace vracející čas za pět minut:

```
public class TimeUtilsA {
    public static DateTime InFiveMinutes =>
        DateTime.Now.AddMinutes(5);
}
public class TimeUtilsB {
    public static DateTime InFiveMinutes { get; } =
        DateTime.Now.AddMinutes(5);
}
```

(A) Jsou obě implementace funkčně ekvivalentní nebo jsou mezi nimi nějaké rozdíly? Vysvětlete a zhodnoťte, která je vhodnější.
[1 bod]

(B) U struktury `DateTime` jsme si všimli, že metody jako výše použitá `AddMinutes` vždy vracejí novou instanci této struktury jako návratovou hodnotu, a nikdy nemodifikují stávající. Stejně tak vlastnosti jako `Minute` nebo `Second` mají na struktuře pouze getter. Vysvětlete, zda to je jen opomenutí autorů standardní knihovny .NET, nebo zda má tento způsob implementace nějaké výhody (a rozveďte jaké).
[1 bod]

Otázka č. 4

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

```
class Prg4 {
    static void Func<X>(X v) {
        Console.WriteLine("1");
    }

    static void Func(int i) {
        Console.WriteLine("2");
    }

    public static void SafeFunc<X>(X value) {
        if (value == null)
            throw new ArgumentNullException(
                nameof(value)
            );
        Func(value);
    }

    public static void Main(string[] args) {
        SafeFunc(123);
        SafeFunc(3.14);
    }
}
```

[1 bod]

Otázka č. 5

Vysvětlete v kontextu jazyka C# a platformy .NET termíny *verifiable code*, *unverifiable code*, a *unsafe code*. Jaký je mezi nimi vztah?
[1 bod]

Otázka č. 6

Předpokládejte následující program:

```
class A {
    public virtual int m(int x) {
        return x - 10;
    }
}

class B : A {
    public virtual int m(int x) {
        return x + 10;
    }
}

class C : B {
    public override int m(int x) {
        return base.m(x) * 2;
    }
}

class Prg6 {
    public static void Main() {
        A a = new C();
        Console.WriteLine($"{a.m(1)}");
    }
}
```

(A) Co tento program vypíše na standardní výstup? Vysvětlete proč.
[1 bod]

(B) Lišilo by se nějak chování programu, kdyby ve třídě `B` byla metoda `m` uvozena klíčovým slovem `override` místo `virtual`? Pokud ano, tak vysvětlete, proč jazyk C# umožňuje v této situaci volbu mezi těmito možnostmi. Pokud ne, tak vysvětlete, proč jsou v jazyce C# obě klíčová slova `override` i `virtual`, a zda by v jiné situaci byl mezi nimi nějaký rozdíl.
[1 bod]

Otázka č. 7

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

```
class Prg7 {
    public static void Main() {
        int a = 5;
        var x = $"a is {a}";
        a++;
        Console.WriteLine(x);
    }
}
```

[1 bod]