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).

Společná část pro otázky označené A

Předpokládejte následující program:

```
A1 a1 = new B1();
Console.WriteLine(a1.f());

class A1 {
  public int f() => 1;
}
class B1 : A1 {
  public new int f() => 2;
}
```

Otázka č. 1 (A)

Co uvedený program vypíše? Detailně vysvětlete proč. Vysvětlete, kdo a jak (C# překladač, nebo JIT, nebo přímo výsledný strojový kód za runtimu) danou variantu zvolí.

Otázka č. 2 (A)

Uvedený program beze změny funguje v projektu vytvořeném ve VS 2022 ze šablony *Console App* cílící na .NET 7, nebo při vytvoření projektu na příkazové řádce pomocí `dotnet new console`. V ReCodExu se ale program nepřeloží, a je třeba na začátek zdrojového souboru dopsat řádek:

```
using System;
```

Vysvětlete, čím je chování v ReCodExu asi způsobeno, zda by bylo možné podobné chování (vynucení `usingu`) docílit i v projektech vytvořených některým z výše uvedených způsobů.

Společná část pro otázky označené B

Předpokládejte, že programujeme karetní hru v kontextu magického světa. Uvedená třída `MagicCard` reprezentuje herní kartu s mnoha různými vlastnostmi (pro přehlednost vynechané). Metoda `Create` slouží k načtení informací o konkrétní kartě z „databáze“ vytvořené designery hry:

```
class MagicCard {
  private static HttpClient s_httpClient
    = new HttpClient();

  /// <param name="cardName">Standard English name
  /// of a card from any supported deck,
  /// e.g. "Gixian Recycler",
  /// or "Tomakul Phoenix"</param>
  /// <exception cref="HttpRequestException">
  /// A card with cardName name is not
  /// supported.</exception>
  /// <exception cref="JsonException">
  /// Card failed to load due to invalid
  /// format.</exception>
  public MagicCard Create(string cardName) {
    var lowerCase = cardName.ToLower();
    var json = s_httpClient.GetStringAsync(
      $"http://d3s.mff.cuni.cz/~jezek/{
        lowerCase
      }.json").Result;
    var card =
JsonSerializer.Deserialize<MagicCard>(json);
    return card;
  }
}
```

Otázka č. 3 (B)

Naše aplikace bude běžet přímo na počítačích hráčů s implicitním nastavením. Je způsob získávání jména souboru s kartou z jejího jména nejvhodnější? Detailně vysvětlete, zda a proč ano, resp. zda a proč ne. Pokud stávající řešení není vhodné, tak danou část C# kódu přepište na vhodnější.

Otázka č. 4 (B)

Část API funkce `Create` je popsána v dokumentačních komentářích – je tento metodou slibovaný kontrakt navržen vhodně vzhledem k budoucí rozšiřitelnosti a udržovatelnosti kódu? Pokud ano, tak vysvětlete proč, pokud ne, tak navrhněte jeho úpravu (a případně adekvátně upravte C# kód metody), a vysvětlete výhody vašich úprav.

Otázka č. 5

Detailně vysvětlete, zda je vhodnější následující typ `Color` nechat deklarovaný jako `class`, nebo zda by bylo vhodnější ho deklarovat jako `struct`. Jaké jsou výhody vámi zvolené varianty, a jaké nevýhody varianty opačné?

```
class Color {
  public byte R;
  public byte G;
  public byte B;
}
```

Společná část pro otázky označené C

Předpokládejte následující program:

```
C6 c6 = new D6();
Console.WriteLine(c6.f());
I6 i6 = new C6();
Console.WriteLine(i6.f());

interface I6 {
  public char f();
}
class A6 {
  public virtual char f() => 'A';
}
class B6 : A6, I6 {
  public override char f() => 'B';
}
class C6 : B6 {
  public virtual char f() => 'C';
}
class D6 : C6 {
  public override char f() =>
    (char) (base.f() + 1);
}
```

Otázka č. 6 (C)

Co uvedený program vypíše na 1. řádku výstupu? Detailně vysvětlete proč – jako součást vysvětlení nakreslete VMT tříd `A6`, `B6`, `C6`, `D6`.

Otázka č. 7 (C)

Co uvedený program vypíše na 2. řádku výstupu? Detailně vysvětlete proč – jako součást vysvětlení nakreslete interfacovou tabulku metod pro interface `I6` u tříd `A6`, `B6`, `C6`, `D6`.

Společná část pro otázky označené D

Předpokládejte následující program:

```
enum StudyProgram {
  Mathematics, Physics, ComputerScience
}

class Person {
  public string Name { get; init; }
  public StudyProgram Program { get; init; }
}

class Prg8 {
  public static void Main() {
    var jan = new Person {
      Name = "Jan",
      Program = StudyProgram.Physics
    };

    PrintInfo(jan);
    PrintInfo("Pavel");
  }

  private static void PrintInfo(
    object personOrName
  ) {

    var info = personOrName switch {
      Person(var name, isComputerScientist: true)
        => $"nerd {name}",
      Person(var name, isComputerScientist: false)
        => $"student {name}",
      string name => name
    };

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

Otázka č. 8 (D)

Uvedený program se nepřeloží, a překladač na označených řádcích vypíše následující chybu:

```
Error: 'Person' does not contain a definition for 'Deconstruct'
```

Napište kompletní implementaci kódu chybějícího ve třídě `Person` tak, aby se uvedený program přeložil.

Otázka č. 9 (D)

Na uvedeném příkladu vysvětlete, co je tzv. *„duck typing“*. Využijte uvedený příklad pro vysvětlení výhod i nevýhod duck typingu v kontextu jiných možných přístupů.

Otázka č. 10

Předpokládejte níže uvedenou třídu `Image` pro reprezentaci bitmapových obrázků, a její potomky pro načítání a ukládání obrázků z formátů JPEG a BMP. Ve třídě `Prg10` naleznete základní příklad použití daných tříd. Vysvětlete, proč tento objektový návrh není vhodný, a upravte ho na vhodnější (včetně adekvátní úpravy metody `Main`). Vysvětlete, v čem je váš objektový návrh lepší.

Metoda `UpdateImageForProjector` by měla zůstat beze změn.

```
abstract class Image {
  protected Color[,]? bitmap = null;

  public void ChangeBrightness(float amount) {
    // Implementace změny světlosti
  }

  public void ChangeContrast(float amount) {
    // Implementace změny kontrastu
  }

  public abstract void Load(string fileName);
  public abstract void Save();
}

class JpegImage : Image {
  public override void Load(string fileName) {
    // Načítání JPEG obrázku do pole bitmap
  }
  public override void Save() {
    // Ukládání JPEG obrázku z pole bitmap
    // do soouboru zapamatovaného jména
  }
}

class BmpImage : Image {
  public override void Load(string fileName) {
    // Načítání BMP obrázku do pole bitmap
  }
  public override void Save() {
    // Ukládání BMP obrázku z pole bitmap
    // do souboru zapamatovaného jména
  }
}

class Prg10 {
  public static void Main() {
    var image = new JpegImage();
    image.Load("blackboard.jpg");
    UpdateImageForProjector(image);
    image.Save();
  }

  public static void UpdateImageForProjector(
    Image image
  ) {
    image.ChangeBrightness(1.75f);
    image.ChangeContrast(1.6f);
  }
}
```