

1. Követelmények

- A feladat kiírásnak megfelelő megvalósításával 18 pontot lehet szerezni.
- A megoldást tömörítve kell feltölteni NEPTUNKÓD.zip formátumban a feladatnak létrehozott beadási felületen.
- Az archívum tartalmazza az egész Solution-t (.sln és .csproj fájlokat is), azaz kicsomagolás után betölthető és futtatható legyen Visual Studio-ban.
- Tömörítés előtt futtasd le a *dotnet clean* parancsot, a bin és obj mappák törölendők és a kiadott CSV-ket se tartalmazza a tömörített állomány.
- A program a .NET 8-as verzióját használja.
- A beadott megoldásnak fordulnia és futnia kell, a projekt mappában a *dotnet run <argumentumok>* parancs hatására induljon el az alkalmazás, ellenkező esetben a projektmunkára automatikusan 0 pont jár.
- Részpontoszám szerezhető, azaz ha nem sikerül valamely feladatot teljes mértékben megoldani, attól nem kell kitörölni (de forduljon és fusson a projekt).
- A kiadott CSV-k tartalmazhatnak hibákat. A cél nem a CSV hibáinak kézzel történő javítása, így arra nem jár pont. A kiértékelés a kiadott CSV-vel történik.
- Azokban az esetekben, ahol a feladatkiírás nem ad konkrét iránymutatást vagy részletezést, a megoldás módja a hallgató saját belátására és kreativitására van bízva.

2. Technikai segédlet

A fejlesztőkörnyezetben a parancssori argumentumok beállítása a következő módon történik:

- Solution Explorer-ben jobb kattintás a projekt nevére -> Properties
- Debug fül, majd General
- Open debug launch profiles UI -> Command line arguments

```
static void Main(string[] args) {  
    foreach (var arg in args)  
        Console.WriteLine(arg); // 1 2 3  
}  
// dotnet run 1 2 3
```

CSV fájlok beolvasására használható a CSVHelper¹ NuGet csomag. CLI-ből a *dotnet add package CsvHelper --version 33.1.0*

parancs kiadásával telepíthető, vagy Visual Studio-ból a *Manage NuGet packages* menüpont alól.

¹<https://joshclose.github.io/CsvHelper/getting-started/>

3. Feladatok

1. Készítsd egy platformfüggetlen parancssoros .NET 8 alkalmazást, amely parancssori argumentumként várja az *athletes.csv* és a *countries.csv* fájlok elérési útvonalait. A program pontosan annyi CSV-t várjon amennyit a feladat említ, minden más esetben jelezze a hibát.

1 pont

2. Készítsd el a fájlokban található adattípusoknak megfelelő osztályokat külön fájlokban, property-k használatával. A program futása során az adatok csak egyszer kerüljenek betöltésre egy általad választott kollekcióba.

4 pont

3. LINQ segítségével határozd meg az összes sportoló átlagéletkorát.

Elvárt eredmény: szám (2 tizedesjegyre kerekítve)

1 pont

4. LINQ segítségével keresd meg, majd kistázd ki a legmagasabb sportoló nevét és magasságát.

Elvárt eredmény: "név: magasság cm", pl. "John Doe: 200 cm"

2 pont

5. LINQ segítségével számold meg, hogy csapatonként hány sportoló szerepel az adathalmazban.

Elvárt eredmény: "csapat név: számosság" egymás után kiírva a számosság alapján csökkenő sorrendben, pl. "Hungary: 12"

Fontos: Ne a csapat rövidítését írd ki, hanem a teljes nevét (HUN →Hungary).

2 pont

6. LINQ segítségével határozd meg a női sportolók átlagmagasságát.

Elvárt eredmény: "Női sportolók átlagmagassága: X cm (2 tizedesjegyre kerekítve)".

2 pont

7. LINQ segítségével keresd meg, majd listázd ki azoknak a sportolóknak a nevét ABC sorrendben, akik 25 év alattiak.

Elvárt eredmény: "Név" egymás után kiírva a megfelelő sorrendben.

2 pont

8. LINQ segítségével minden csapatból keresd meg a legnehezebb sportolót. A kiíratás a legnehezebb sportolótól induljon a legkönnyebb felé.

Elvárt eredmény: "csapat teljes neve: sportoló neve (tömeg kg)" egymás után kiírva a megfelelő sorrendben.

2 pont

9. LINQ segítségével csoportosítsd a sportolókat nem szerint, és számítsd ki mindkét csoport átlag-életkorát.

Elvárt eredmény: "nem: átlagéletkor (2 tizedesjegyre kerekítve)" egymás után kiírva.

2 pont

A feladatok eredményeit az STDOUT-ra írd ki a következő formában:

X. feladat:

megoldás

[megoldás] (amennyiben több adatot kell kiírni, soronként)