.NET FEJLESZTÉS C# NYELVEN

4. forduló

DXC TECHNOLOGY

A kategória támogatója: DXC Technology

Ismertető a feladathoz

A 4. forduló után elérhetőek lesznek a helyezések %-os formában: azaz kiderül, hogy a kategóriában a versenyzők TOP 20% - 40% -60% -ához tartozol-e!

Szeretnénk rá felhívni figyelmedet, hogy a játék nem Forma-1-es verseny! Ha a gyorsaságod miatt kilököd a rendesen haladó versenyzőket, kizárást vonhat maga után!

Felhasznált idő: 06:28/10:00 **Elért pontszám:** 0/15

1. feladat 0/5 pont

Mi lesz a következő kód kimenete?

Ugrás a tartalomhoz

```
public class Sample
{
    public void Run()
    {
        var collection = InitializeCollection();

        Console.WriteLine("Collection Initialized");
        Console.WriteLine($"Sum: {collection.Sum()}");
    }

    private IEnumerable<int> InitializeCollection()
    {
        yield return 1;
        Console.WriteLine("Added '1'");
        yield return 2;
        Console.WriteLine("Added '2'");
    }
}
```

Válasz

- ○ Added '1'

  Added '2'

  Collection Initialized

- ● Added '1'

  Added '2'

  Collection Initialized

  Sum: 3

  Ez a válasz helytelen, de megjelölted.

- ◉ Collection Initialized

  Added '1'

  Added '2'

  Sum: 3

  Ez a válasz helyes, de nem jelölted meg.

- ○ Added '1'

  Added '2'

  Collection Initialized

  Sum: 2

Ugrás a tartalomhoz

Magyarázat

2. feladat 0/10 pont

Adott az alábbi *AsyncWorker* osztály és annak használata. Vizsgáld meg a hozzájuk tartozó állításokat!

```
internal class AsyncWorker
{
    public async Task DoSomething(string tag)
    {
        Console.WriteLine($"{tag} before await");
        await Task.Delay(500);
        Console.WriteLine($"{tag} after await");
    }
}

var worker = new AsyncWorker();
var tasks = new Task[]
{
    worker.DoSomething("cheese"),
    worker.DoSomething("goat")
};
await Task.Delay(1000);
Console.WriteLine("Before WaitAny");
Task.WaitAny(tasks);
Console.WriteLine("After WaitAny");
```

Állítások

1. Biztosan először a "cheese worker" majd a "goat worker" fog lefutni, mert a *Task.WaitAny* fogja elindítani őket.
2. A "goat worker" biztosan hamarabb végez a *"Before WaitAny" kiírásánál*, mert a Compailer kioptimalizálja a kódot.

Válaszok

Ugrás a tartalomhoz

Az első állítás első tagmondata igaz.

Ez a válasz helytelen, de megjelölted.

- [x] Az első állítás második tagmondata igaz.
  Ez a válasz helytelen, de megjelölted.
- [x] Az első állítás mindkét tagmondata igaz.
  Ez a válasz helytelen, de megjelölted.
- [x] Az első állítás mindkét tagmondata hamis.
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Az második állítás első tagmondata igaz.
- [x] Az második állítás második tagmondata igaz.
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Az második állítás mindkét tagmondata igaz, helytelen a következtetés.
- [ ] Az második állítás mindkét tagmondata igaz, helyes a következtetés.
- [ ] Az második állítás mindkét tagmondata hamis.

Magyarázat

A lefutás sorrendje nem determinisztikus. Nem *Task.WaitAny* indítja el őket. Compailer igyekszik mindig kioptimalizni a kódot.