IPAR 4.0 C# .NET ALAPOKON

4. forduló

SEMILAB

A kategória támogatója: Semilab Zrt.

RENDELKEZÉSRE ÁLLÓ IDŐ:

45:00

Ismertető a feladathoz

Fontos információk

A forduló után a megoldások publikálásával együtt iránymutatásként elérhetőek lesznek a **helyezéssel kapcsolatos információk**, látni fogod, hogy a kategóriában a játékosok 20%, 40% vagy 60%-a közé tartozol-e épp.

Felhívjuk figyelmedet, hogy a következő, **5. fordulótól az egyes kategóriák csak a kijelölt napokon lesznek megoldhatóak 7-22 óra között**, érdemes letöltened a naptárat a Kategóriáim menüpontban.

Negyedik forduló

A gyár további folyamatok elvégzéséhez kéri a segítségünket, részletes leírások a feladatokban.

Felhasznált idő: 02:07/45:00 **Elért pontszám:** 0/7

1. feladat 0/2 pont

Mit ír ki az alábbi kódrészlet?

```
6 references
internal struct MyStruct
{
    5 references
    public int N { get; set; }
    4 references
    public MyStruct(int n)
    {
        N = n;
    }
    2 references
    public void Increment(int amount)
    {
        N += amount;
    }
}

0 references
private static void Main(string[] args)
{
    var variable = new MyStruct(0);
    variable.Increment(1);
    var list = new List<MyStruct>
    {
        new MyStruct(0),
        new MyStruct(1),
        new MyStruct(2)
    };

    foreach (var item in list)
    {
        item.Increment(variable.N);
    }

    Console.WriteLine(list.Sum(item => item.N) + variable.N);
}
```

Válaszok

A helyes válasz:
4

Magyarázat

A Struct típus a stacken jön létre. A variable lokális változónk is a stacken jön létre, majd növekszik 1-el. A List típus a heapen jön létre, a benne lévő Struct objektumok a heapen vannak becsomagolva. A foreach ciklusnál az item lokális változó a stacken jön létre, a heapen lévő listaelemek egyesével belemásolódnak. Amikor az item változón hívjuk az Incrementet, igazából csak a másolatot növeljük, a listaelem változatlan marad. Tehát a listaelemeink értéke 0,1,2 marad, míg a variable 1 lesz. Így jön ki a 4-es összeg. Ha Struct helyett Classként definiáltuk volna a típust, akkor más működést kapnánk.

2. feladat 0/5 pont

Egy minőség-ellenőrzésért felelős mérőgép különböző tulajdonságokat vizsgál egy adott mintán. Egy tulajdonság vizsgálatához több tesztet is futtathat. Egy adott tulajdonságot egy előjeles összeg minősít, melybe a sikeres tesztek +1, a sikertelen tesztek -1 értékkel számítanak bele.

A vizsgált tulajdonságokat az angol abc egy betűje jelöli. A mérőgép kimenete egy sztring, melyben kis- és nagybetűk szerepelnek. Az adott betűk a tulajdonságoknak felelnek meg, a nagybetűk jelentik, hogy a teszt sikeres volt, a kisbetűk pedig, hogy sikertelen.

Példa bemenet:

AFab

Ez azt jelenti, hogy az első teszt az "a" tulajdonságot vizsgálta és sikeresen teljesült, a második az "f" tulajdonságot és szintén sikeres volt, a harmadik ismét az "a" tulajdonságot és sikertelen volt, míg a negyedik a "b" tulajdonságot és sikertelen volt.

A feladat az, hogy minden vizsgált tulajdonsághoz kiszámítsuk a tulajdonságot minősítő előjeles összeget és ezeket **abc sorrend alapján**, **[tulajdonság betűjele kisbetűvel]:[előjeles összeg integerként]** formátumban, vesszővel elválasztva kiírjuk.

A fenti példához a helyes kimenet az alábbi lenne**:**

a:0,b:-1,f:1

Magyarázat: "a" tulajdonsághoz volt egy sikeres és egy sikertelen teszt, tehát az összeg 0, "b"-ből egy sikertelen volt, így az összeg -1, míg "f"-ből egy sikeres, tehát az összeg 1.

Adjuk meg a **4_fordulo_2_feladat.txt** fájlban található bemenethez tartozó kimenetet!

Teszteléshez a 4_fordulo_2_feladat_test.txt kimenete: a:-11,d:3,f:-12,s:-13

Válaszok

A helyes válasz:
a:-11,d:-31,f:27,s:-27
"a:-11,d:-31,f:27,s:-27"
a:-11, d:-31, f:27, s:-27
"a:-11, d:-31, f:27, s:-27"
a: -11,d: -31,f: 27,s: -27

Magyarázat

```
private static string GetTestResults(string fileName)
{
    var s = File.ReadAllText(fileName);
    Dictionary<string, int> dict = new Dictionary<string, int>();
    foreach (char c in s)
    {
        string key = c.ToString().ToLower();
        if (!dict.ContainsKey(key))
        {
            dict.Add(key, 0);
        }
        if (char.IsUpper(c))
        {
            dict[key]++;
        }
        else
        {
```

```
            dict[key]--;
        }
    }

    List<string> keys = dict.Keys.OrderBy(d => d).ToList();

    IEnumerable<string> values = keys.Select(k => $"{k}:{dict[k]}");
    return string.Join(",", values);
}
```