



A feladat megoldását a Program.cs fájlba készítse el, melyet beadás előtt nevezzen át. A beadandó forrásfájl elnevezése a feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD.cs**

A feladattal kapcsolatos további információk az utolsó oldalon találhatóak (ezen ismeretek hiányából adódó reklamációt nem fogadunk el!).

A 7 szegmenses kijelzőn ábrázolt ASCII Art formátumú számsorozatot kell számokká alakítani.

Bemenet (Console)

- három sor, ami leírja a számsorozatot (minden sor azonos számú karaktert tartalmaz)

Kimenet (Console)

- a bemeneti számsor számként megjelenítve

Megkötések

- $3 \leq$ a kijelző digitjeinek a száma ≤ 1000
- a bemenet minden I karakterére: $I \in \{ |, -, szóköz \}$
- a bemeneti digitek csak számokat írnak le
- egy digit 3×3 egység
- a kimenetnek a sor elején lévő vezérlő 0-kat is tartalmaznia kell
- a Program osztály mellett pontosan 1 darab osztályt kell implementálni (ugyan abban a fájlban mint ahol a Program osztály található), ami megvalósítja az átalakítást
- a Program osztály csak a Main metódust tartalmazhatja, a Main metódus a 3 sor beolvasását, a saját osztály példányosítását és az átalakítást megvalósító metódusának meghívását, illetve az eredmény kiírását tartalmazhatja

Megjegyzés

- Amennyiben van rá lehetősége .NET Core helyett .NET Framework alkalmazás részeként készítse el a megoldást.

Példa

Console input
1 _ _ _ _ _ _ _ _
2 _ _ _
3 _ _ _

Console output
1 0123456789

Értelmezés

A bemenet 3 sort tartalmaz, aminek az első digitje egy 0. Következő 3×3 -as blokk alapján az 1-est kell kiírni és így tovább, míg az összes digitet nem dolgozzuk fel és jelenítjük meg a hozzá tartozó számokat a kimeneten. Tehát a megjelenítendő elkódolt üzenet a következő: 0123456789

Teszt esetek

Az alkalmazás helyes működését legalább az alábbi bemenetekkel tesztelje le!

1.

Console input
1 _ _ _ _ _ _ _ _
2 _ _ _
3 _ _ _

Console output
1 0123456789



2.

```
Console input _____
1  _ _ _
2  | | | |
3  | _ | _ |
```

```
Console output _____
1  000
```

3.

```
Console input _____
1  _ _ _
2  | _ _ |
3  | _ | _ |
```

```
Console output _____
1  625
```

4.

```
Console input _____
1  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2  | | | | | | | | | | | | | | | | | |
3  | _ | _ | _ | _ | _ | _ | _ | _ | _ |
```

```
Console output _____
1  0000000123456789000000
```

5.

```
Console input _____
1  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
2  | | | _ | _ | _ | _ | _ | _ | _ | _ |
3  | _ | _ | _ | _ | _ | _ | _ | _ | _ |
```

```
Console output _____
1  0123456789876543210
```

6.

```
Console input _____
1  _ _ _ _ _
2  | _ | | | _ |
3  | _ | _ | _ |
```

```
Console output _____
1  80085
```

A fenti tesztesetek nem feltétlenül tartalmazzák az összes lehetséges állapotát a be- és kimenet(ek)nek, így saját tesztekkel is próbálja ki az alkalmazás helyes működését!

Tájékoztató

A feladattal kapcsolatosan általános szabályok:

- A feladat megoldásaként beadni vagy a betömörített solution mappa egészét vagy a Program.cs forrásfájlt kell (hogya pontosan melyiket, azt minden feladat külön definiálja), melynek elnevezése a feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD**[.zip|.cs]
- A megvalósítás során lehetőség szerint alkalmazza az előadáson és a laboron ismertetett programozási tételeket és egyéb algoritmusokat.
- Az alkalmazás elkészítése során mindenesetben törekedjen a megfelelő típusok használatára, illetve az igényes (*formázott, felesleges változóktól, utasításoktól mentes*) kód kialakítására, mely magába foglalja az elnevezésekkel kapcsolatos ajánlások betartását is (**bővebben**).
- A megoldásokhoz nem használhatók a beépített rendezőmetódusok (például: `Array.Sort`), a LINQ technológia (`System.Linq`), kivételkezelés (`try-catch-finally` blokk), a `goto`, a `continue` és a `break` (kivéve a `switch-case` szerkezetnél) utasítások, az alábbi gyűjtemények: `ArrayList`, `List`, `SortedList`, `Dictionary`, `Stack`, `Queue`, `Hashtable`, a `var` az `object` és a `dynamic` kulcsszavak, illetve figyelembe kell venni a *Megkötések* pontban meghatározott további szabályokat.
- A leadott feladat megoldással kapcsolatos minimális elvárás a leírásban feltüntetett tesztesetek helyes futtatása, a *Megkötések* pontban definiáltnak való megfelelés, ezeket leszámítva viszont legyen kreatív a feladat megoldásával kapcsolatban.
- A kiértékelés során csak a *Megkötések* pont szerinti helyes bemenettel lesz tesztelve az alkalmazás, a "tartományokon" kívüli értéket nem kell lekezelnie az alkalmazásnak.
- **Ne másoljon vagy adja be más megoldását!** Minden ilyen esetben az összes (felépítésben) azonos megoldás duplikátumként lesz megjelölve, melyek közül kizárólag, az időrendben elsőnek leadott lesz elfogadva.
- **Idő után leadott vagy helytelen elnevezésű megoldás vagy a kiírásnak nem megfelelő megoldás vagy fordítási hibát tartalmazó vagy (helyes bemenetet megadva) futásidejű hibával leálló kód nem értékelhető!**
- A feladat leírása az alábbiak szerint épül fel (* - opcionális):
 - *Feladat leírása* - a feladat megfogalmazása
 - **Bemenet* - a bemenettel kapcsolatos információk
 - **Kimenet* - az elvárt kimenettel kapcsolatos információk
 - *Megkötések* - a bemenettel, a kimenettel és az algoritmussal kapcsolatos megkötések, melyek figyelembevétele és betartása kötelező, továbbá az itt megfogalmazott bemeneti korlátoknak a tesztek minden esetben eleget tesznek, így olyan esetekre nem kell felkészülni, amik itt nincsenek definiálva
 - **Megjegyzések* - további, a feladattal, vagy a megvalósítással kapcsolatos megjegyzések
 - *Példa* - egy példa a feladat megértéséhez
 - *Tesztesetek* - további tesztesetek az algoritmus helyes működésének teszteléséhez, mely nem feltétlenül tartalmazza az összes lehetséges állapotát a be- és kimenet(ek)nek
- **Minden esetben pontosan azt írja ki és olvassa be az alkalmazás, amit a feladat megkövetel, mivel a megoldás kiértékelése automatikusan történik!** Így például, ha az alkalmazás azzal indul, hogy kiírja a konzolra a *"Kérem az első számot:"* üzenetet, akkor a kiértékelés sikertelen lesz, a megoldás hibásnak lesz megjelölve, ugyanis egy számot kellett volna beolvasni a kiírás helyett.
- A kiértékelés automatikusan történik, így különösen fontos a megfelelő alkalmazás elkészítése, ugyanis amennyiben nem a leírtaknak megfelelően készül el a megoldás úgy kiértékelése sikertelen lesz, a megoldás pedig hibás.
- Az automatikus kiértékelés négy részből áll:
 - Unit Test-ek - az alkalmazás futásidejű működésének vizsgálatára
 - Szintaktikai ellenőrzés - az alkalmazás felépítésének vizsgálatára
 - Duplikációk keresése - az azonos megoldások kiszűrésére
 - Metrikák meghatározása - tájékoztató jelleggel
- A kiértékelésnek eredményéből egy HTML report generálódik, melyet minden hallgató megismerhet.