Házi Feladat

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 találhatók (ezen ismeretek hiányából adódó reklamációt nem fogadunk el!).

Szabadúszóként egy házszámozással foglalkozó vállalkozás szoftverfejlesztési projektjét vállalja el.

A vállalkozás minden reggel megkapja, hogy melyik utcában melyik házszámtól (S), hány darab (P) házszámot kell lecserélnie az adott oldalon (melyet a kezdő házszám paritása határoz meg). Mivel a fémáruház csak számjegyeket árusít, illetve, mivel a vállalkozó szeretné minimalizálni a költségeit, így pontosan annyi számjegyet szeretne venni, amennyi szükséges az adott napi munka elvégzéséhez.

Tehát, írjon programot, ami meghatározza, hogy minimálisan hány darab szükséges az egyes számjegyekből (amelyikből egyetlen darab sem kell, ott 0-val jelölje azt), majd jelenítse meg a felhasználónak vesszővel elválasztva az eredményt az alábbi formában (ahol az első érték a 0-ás, a második az 1-es, az utolsó pedig a 9-es számjegyek számát jelentse):

`0,0,0,0,0,0,0,0,0,0`

Bemenet (Console)

- *első értéke* - a kezdő ház száma (S)
- *második értéke* - a lecserélendő házszámok száma (P)

Kimenet (Console)

- a `Console`-on megjeleníve a 10 darab szám vesszővel elválasztva

Megkötés(ek)

- $1 \leq S, P \leq 1000$

Példa

Console input
```
7
3
```

Console output
```
0,2,0,0,0,0,0,1,0,1
```

Értelmezés

A felhasználó a következő kezdő házszámot adja meg: `S=7`

A felhasználó `P=3` házszámot szeretne lecseréltetni (a kezdő ház számát is beleértve).

Mivel a kezdő házszám a `7`, ezért a páratlan oldalon kell lecserélni a házszámokat. Az értintett házszámok: `7, 9, 11`. Mint látható az `1`-es számjegyből kettő darab szükséges, a `7`-es és `9`-es számjegyekből pedig egy-egy.

Végezetül az alkalmazás a következőt jeleníti meg a felhasználónak: `0,2,0,0,0,0,0,1,0,1`

Tesztesetek

Az alkalmazás helyes működését legalább az alábbi bemenetekkel tesztelje le!

A felhasználó által megadott bemenetek		**A bemenethez tartozó elvárt kimeneti értékek**
S	P	
7	3	0,2,0,0,0,0,0,1,0,1
2	4	0,0,1,0,1,0,1,0,1,0
555	1	0,0,0,0,0,3,0,0,0,0
1000	10	17,15,2,0,2,0,2,0,2,0
150	37	13,30,22,0,7,5,12,5,12,5
62	655	246,351,296,151,246,115,250,116,246,115
918	65	42,30,23,10,22,5,18,5,18,46
268	491	198,225,189,100,198,95,194,100,199,100
491	268	43,97,28,79,30,129,75,128,75,133
1	1	0,1,0,0,0,0,0,0,0,0
1000	1000	400,700,900,200,400,200,400,200,400,200

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 (hogy 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 minden esetben 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`, `Hastable`, a `var` az `object` és a `dynamic` kulcsszavak, illetve figyelembe kell venni a *Megkötések* pontban meghatározott további szabályokat.
- A leadott feladat megoldásával kapcsolatos minimális elvárás a leírásban feltüntetett tesztesetek helyes futtatása, a *Megkötések* pontban definiáltaknak 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 eseteben 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 eseteben 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 kiértékelések eredményéből egy HTML report generálódik, melyet minden hallgató megismerhet.