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!).

Írjon programot, amely 1-től a felhasználótól bekért pozitív egész számig (N) meghatározza a számok osztóinak összegét ($DIVSUM$), majd az eredményt megjeleníti a felhasználónak a konzolon az alábbi minta szerint formázva:

```
1-N-ig a számok osztóinak összege:  DIVSUM.
```

Megkötések

- $1 \leq N \leq 99\ 999\ 999$
- az alkalmazásban nem használható:
  - metódusok: `Console.ReadKey`

Megjegyzés

- törekedjen olyan algoritmus elkészítésére, mely futásideje $O(N) \longrightarrow$ ne optimalizáljon, úgy se lesz eredménye, váltson paradigmát

Példa

Console input

```
7
```

Console output

```
1-7-ig a számok osztóinak összege: 41.
```

Értelmezés

A felhasználó a következő számokat adja meg: `N=7`

1-től a megadott számig a számok osztói:

```
1:  1              4:  1, 2, 4          6:  1, 2, 3, 6
2:  1, 2           5:  1, 5             7:  1, 7
3:  1, 3
```

A számok osztóinak összege:

```
DIVSUM: 41 = 1 + 1 + 2 + 1 + 3 + 1 + 2 + 4 + 1 + 5 + 1 + 2 + 3 + 6 + 1 + 7
```

Végezetül az alkalmazás a következőt jeleníti meg a felhasználónak:

```
1-7-ig a számok osztóinak összege:  41.
```

Tesztesetek

Az alkalmazás helyes működését legalább az alábbi bemenetekkel tesztelje le!

A felhasználó által megadott bemenet N	**A bemenethez tartozó elvárt kimeneti értékek** $DIVSUM$
7	41
17	238
1000	823081
90000	6662106690
20170901	334633255156090
99999999	8224670172682646

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 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`, `Hastable`, 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 definiáltaknak való megfelelés, ezeket leszámítva viszont legyen kreatív a feladat megoldásával kapcsolatban.
- A kiértékelés során csak a *Megkötések* pont szerenti 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 megköve-tel, 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.