Programozás Alapjai 6. házi feladat

1. feladatsor

Szoftverfejlesztés Tanszék

2023, Ősz

Feladat Töltsd le a bíróról a `minta.zip` állományt, majd tömörítsd ki! A `feladat.c` fájlban megtalálod a feladatok megoldás-kezdeményeit. Bővítsd ezt az alább olvasható feladatok alapján! Lehetőség szerint ellenőrizd megoldásod, majd töltsd fel a `feladat.c` fájlt a bíróra!

Kiértékelés A bíró lefordítja a programot, majd lefuttatja azt a feladat pontszámának megfelelő számú tesztesettel. Egy teszteset egy bemenet-kimenet pár, amely a megfelelő feladathoz készült. A teszteset akkor helyes, ha az adott bemenethez tartozó kimenet **minden egyes karaktere** megegyezik az előre eltárolt referencia kimenettel. *További feltételek: a program futása nem tarthat tovább 5 másodpercnél, egyszerre nem fogyaszthat többet 16 MiB memóriánál és nem történhet futási hiba (pl. illetéktelen memória hozzáférés).*

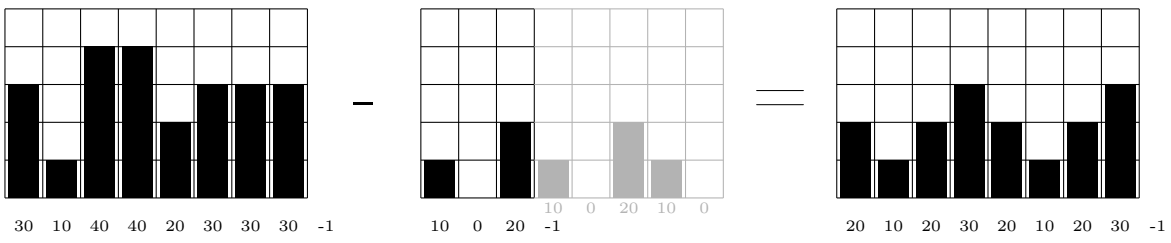
Ellenőrzés Feltöltés előtt érdemes ellenőrizni a megoldásod.

1. **Fordítás** Ellenőrizd, hogy a programod lefordul-e! A bíró a `gcc -O2 -static -o feladat feladat.c` paranccsal fordít, érdemes ezt használni. A `-Wall` kapcsoló is hasznos lehet.
2. **Példa tesztesetek** Ellenőrizd, hogy a programod helyesen működik-e! A `minta.zip` tartalmaz a bíró által futtatott tesztesetek közül feladatonként egyet-egyet. Az első feladat teszteléséhez másold a programod mellé az `ex1.be` fájlt `be.txt` néven, futtasd le a programod, majd az így kapott `ki.txt` tartalmát hasonlítsd össze az `ex1.ki` fájlban található referencia kimenettel.
3. **Extra tesztesetek** Ellenőrizd a programod működését további példák segítségével! Néhány további teszteset is elérhető, de ezek csupán ellenőrzésre használhatóak, a bíró nem futtatja őket. Ezek használatához futtasd a programod a `-t` vagy `-test` kapcsolóval, például a `./feladat -test` paranccsal. Csak az első feladat teszteléséhez futtasd a programod a `./feladat -t 1` paranccsal.

1. feladat (5 pont)
Az ESA csillagászai különös rádiójelet fogtak a világűrből. A jel viszont egy ismert természetes rádióforrás, egy neutroncsillag irányából érkezik, így a két jel összekeveredett. Szerencsére a csillagot régóta figyelik, így pontosan tudják, hogy milyen periodikus jelet bocsájt ki. A csillagászoknak szét kellene választani neutroncsillag ismert periodikus jelét az ismeretlen jeltől.

A fogott jelet csakúgy, mint a neutroncsillag jelét a csillagászok digitalizáló gépei nemnegatív egész számok sorozatává alakítják. A két sorozat különbsége lesz a különös rádiójel digitalizált változata.

Írj függvényt, amely megtisztítja a zajos jelet a neutroncsillag ismert jelétől. A függvény három tömböt kap paraméterként. Az első kettő a `zajos` jel és a `csillag` rádiójele. Mindkettő nemnegatív számok sorozata, amelyet a -1 érték zár (ami viszont már nem része a sorozatnak, csak lezárja azt). A zajos jel nem rövidebb, mint a csillag jele. Utóbbi csak egy periódust tartalmaz a csillag folyamatosan ismétlődő rádiójeléből. A függvény harmadik paramétere szintén egy tömb, amelybe a `tiszta` rádiójelet kell előállítani úgy, hogy a `zajos` tömb adott eleméből kivonjuk a `csillag` tömb azonos sorszámú elemét. (Ha a `csillag` tömb közben „elfogy”, akkor az elejétől kezdve újra felhasználjuk az elemeit, ahányszor csak szükséges.) A tiszta jel szintén csak nemnegatív érékeket fog tartalmazni (ezt nem kell külön ellenőrizni), és a `tiszta` tömböt is a -1 érékkel kell lezárni. (A hossza ugyanaz lesz, mint a `zajos` tömbé.)



Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass! A függvény inputjai és outputja a paraméterek. A függvény nem végez IO műveleteket!

```
void jeltisztitas(int zajos[], int csillag[], int tiszta[]);
```