# Programozás Alapjai 3. próba feladat

**1. feladatsor**

Szoftverfejlesztés Tanszék

2023, Ősz

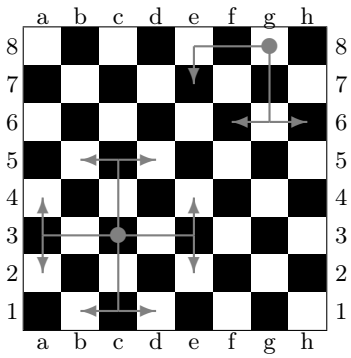
**Feladat** Töltsd le a bíróról a `minta.zip` állományt, majd tömörítsd ki! A `feladat.c` fájlban megtalálod a feladatok megoldás-kezdeményeit. Bővítsd ezt az alább olvasható feladatok alapján! Lehetőség szerint ellenőrizd megoldásod, majd töltsd fel a `feladat.c` fájlt a bíróra!

**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)
A feladat megállapítani, hogy egy huszár hány mezőt üthet a sakktáblán. A huszár a tőle L alakban egyenesen 2 oldalra 1 távolságra lévő mezőket üti (ahol az egyenes bármelyik irány lehet a 4 közül).




A függvény bemenetként karakterben kapja meg a mező koordinátáit (egy kisbetű 'a'-'h' és egy számjegy '1'-'8'), a visszatérési értéke pedig az ütött mezők száma.

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

```
int huszar(char oszlop, char sor);
```

**2. feladat** (5 pont)
Az alábbi függvény paramétere egy sztring. A feladat törölni belőle minden második karaktert. (Ezáltal a sztring rövidülhet.) Kódold le a függvényt C nyelven.

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

```
void strtorles(char str[]);
```

**3. feladat** (5 pont)
Határozzuk meg, hogy a paraméterként kapott évben az adott hónap adott napja az év hányadik napja. A függvény ezzel az értékkel térjen vissza. Figyeljünk a szökőévekre! Az input adatok egy 1800 és 5000 közötti valós napot jelölnek.

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

```
int evnapja(int ev, int ho, int nap);
```

**4. feladat** (5 pont)
Az alábbi függvény egy pénzösszeget kap paraméterként, majd meghatározza, hogy legkevesebb hány darab 10, 5, 2 és 1 koronás érmével fizethető ki az összeg! A függvény visszatérési értéke -1, ha rossz értéket kap.

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

```
int erme(int);
```