

# Programozás Alapjai 9. ZH

## 30. 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 tesztessel. Egy tesztet egy bemenet-kimenet pár, amely a megfelelő feladathoz készült. A tesztet 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 tesztet 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 alábbi függvény feladata helyet foglalni egy kétdimenziós `char` tömbnek. A tömb sor- és oszlopszáma megegyezik. A tömb  $N$  méretét a függvény paraméterként kapja. A helyfoglalás úgy történjen, hogy a kétdimenziós tömb összes elemét egy egydimenziós  $N \times N$ -es tömbben helyezzük el sorfolytonosan. A helyfoglalás után töltsük fel a tömb elemeit értékekkel a következő módon: A tömb páros sor- és oszlopindex-összegű elemei legyenek `'V'`, páratlan indexösszegű elemei pedig `'S'` értékűek. A függvény térjen vissza a tömbre mutató pointerrel. A memória felszabadításával nem kell foglalkoznod.

```
char *foglal(int n);
```

### 2. feladat (5 pont)

A feladat az, hogy egy matematikai kifejezésben eldöntsük, hogy a zárójelezés helyes-e. A kifejezésben a `'(, ')'` zárójelek, `'+', '-', '*', '/'` műveleti jelek, kis és nagybetűk, illetve számok szerepelhetnek. A következő függvény egy sztringet kap paraméterül, melyben az adott kifejezés szerepel. A függvénynek meg kell mondania azt a legkisebb pozíciót, ahol a kifejezésben először előfordul egy zárójelezési hiba. (Ha a hiba egy hiányzó bezáró zárójel, akkor a hiba a sztring végén van.) Ha a kifejezésben nem fordul elő hiba, akkor a függvénynek a `-1` értéket kell visszaadnia! A kifejezés esetleges egyéb hibáival (pl. műveleti jel elmaradása) ne foglalkozunk!

```
int zarojel(char* kifejezes);
```