Programozás Alapjai 9. ZH

14. 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 alábbi függvény feladata helyet foglalni egy kétdimenziós `int` 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: Minden cellába írjuk be a sor- és oszlopindexek összegét. A függvény térjen vissza a tömbre mutató pointerrel. A memória felszabadításával nem kell foglalkoznod.

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

2. feladat (5 pont)
Írj egy függvényt, ami egyszerű tömörítést végez: összevonja a karakterláncban az egymás után többször előforduló ugyanolyan karaktereket, és mögéjük írja, hogy hány darab szerepelt belőlük, ami garantáltan mindenhol egy 2 és 9 közti szám lesz. Az egyszer előforduló karaktereket úgy hagyja, ahogyan voltak. A függvény a második paraméterben állítsa össze az eredményt. Példa: „`eeelemerr belllla`” sztringből „`e3lemer2 bel4a`” lesz.

Write a function that performs simple compression: it combines the same characters that occur several times in a string, and writes behind them how many of them there were, which is guaranteed to be a number between 2 and 9 everywhere. The characters that occur once are left as they were. The function compiles the result in the second parameter. Example: the string „texttteeelemerr belllla” becomes „texttte3lemer2 bel4a”.

```
void betomorit(char* eredeti, char* tomoritett);
```