Programozás Alapjai 4. 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)
Nancy és Jane a következő titkosírással leveleztek A. A. Milne "Micsoda négy nap!" című regényében: Megszámolták, hány betűből állanak a titkosítandó üzenet szavai, majd minden szó betűit véletlenszerűen összezagyválták, leírták folyó írassal, és ebbe a szövegbe szintén sorrendet tartva, de véletlenlenszerű helyekre beszúrták a szavak hosszát jelző számokat.

Például: „*Haho, ez egy uzenet!*" → „*haohzeg4ye2t3nez6eu*".

Valósítsuk meg ennek a titkosírásnak az egyszerűsített verzióját!

A titkosító függvény paramétere két karaktertömb, egy bemenet és egy kimenet. A titkosítandó üzenet a bemenetben van, a titkosítottat a kimenetbe kell elkészíteni. A függvény számolja meg, hogy a bemenet szavai hány betűsek, és ezeket a számokat írja sorrendet tartva a kimenet elejére. Az egyszerűség kedvéért minden bemenet maximum 9 betűs szavakat tartalmaz. Egy szónak számítunk minden karaktert két space között. (Tehát pl. az írásjelek a szóhoz tartoznak. A magányos karakterek, mint pl. a gondolatjel, egybetűs szónak számítanak.) Ez után a függvény a bemenet minden szavát fordítsa meg, és space nélkül, sorrendet tartva írja be a kimenetbe.

Például: `"Haho, ez egy uzenet!"` → `"5237,ohaHzeyge!tenezu"`

```
void titkosit(char bemenet[], char kimenet[]);
```