Programozás Alapjai 4. 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: TITKOS ÜZENET (2 pont)
Az UwU kolónia nagyon titokzatos, és egészen ezidáig nem igazán hallottunk róluk, most azonban megtaláltunk egy rejtett üzenetet a gyönyörűszép kertünkben, ahol a paradicsomok az égig érnek (de ez most lényegtelen).

Az alábbi függvény feladata, hogy ezt a talált üzenetet megismételje, mert mint tudjuk, az ismétlés a tudás anyja. Az üzenet így szól: "uwwu u wwu uww wuwu". A jelentése jelenleg nem ismert, így bármi lehet, de persze nem akármi. A kiíratás végén természetesen egy sortörés is szerepeljen, mert hát sortörés nélkül mit érek én?

A függvény a jól végzett munkát egy 0-s értékkel jelezze, ami nyilván nem azt jelenti, hogy 0 munkát végeztünk. Hát mi vagyunk mi, naplopó informatikusok, akik egész nap nem csinálnak semmit és csak a számítót nyomkodják?

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass!

```
int uzenet();
```

2. feladat: KOLÓNIÁK (3 pont)
UwU kolóniákat találtunk a kertünk gyönyörűszép gyepjén, összesen hármat. A kolóniákban picike uwu lakáskák vannak. A kérdés, amit feltettünk magunknak, hogy összesen mennyi van (a gyanútlan nézelődőben persze felmerülhet a kérdés, hogy minek mászunk a füvön négykézláb, mint valami eltévedt halacska... de mi ismerjük a küldetésünket és hiszünk a csodálatos jövőben).

A függvény feladata, hogy beolvasson három egész számot a standard inputrol, amelyek megmondják, hogy az egyes kolóniákban hány picike uwu lakáskákat találtunk, majd pedig kiírja a standard outputra, hogy hány picike uwu lakás van összesen.

A kiíratás formája az alábbi legyen: "Osszes picike uwu lakaskak szama: <N>.", ahol az <N> helyére az eredmény kerüljön. Természetesen a kiíratást egy soremeléssel zárjuk, továbbá jelezzük az örömteli munka elvégzését egy 0-s visszaadott értékkel.

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass!

```
int lakohely();
```

3. feladat: SPECIÁLIS (4 pont)

Az UwU kolóniák megfigyelése során azt vettük észre, hogy a 3-as szám nagyon nagy szerepet játszik az uwuk életében, ugyanis minden kolóniában lévő picike uwu lakáskák száma 3-ra végződött (pl. 3, 13, 23, 33, stb.).

Persze lehet csak mi képzelődünk a sok fűtől... akarom mondani a sok mászástól, és amúgy semmi összefüggés nincs, de lássuk be, erre elég csekélyke esély van, már nem azért a makadámdióért.

Az alábbi függvény ezúttal paraméterben vár egy értéket, amelyről el kell döntenie, hogy speciális-e, azaz 3-asra végződik-e. Amennyiben speciális számról van szó, akkor írjuk ki a standard outputra, hogy "SPECIALIS", különben írjuk ki, hogy "NEM SPECIALIS". A kiíratást nem meglepő módon egy sorvége jellel zárjuk.

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass! A függvény inputjai a paraméterek.

```
void specialis(int);
```

4. feladat: ZENE (4 pont)

Az UwU kolónia felfedezésére irányuló expedíciónk már nagyon hosszú ideje tart, már kezdünk nagyon kifáradni. De nézzük a jó oldalát, ezen 5 perc alatt már rengeteg felfedezést tettünk, amely alapjaiban változtathatja meg az emberi civilizáció működését. Persze ha nem halunk éhen közben, mert lássuk be, ennek megvan az esélye. Azonban egyik UwU kolóniában egy érdekes dolgot láttunk. Illetve hogy egészen pontos legyek (zümmögnek itt...) hallottunk. Az egyik picike uwu lakáskából csodálatos muzsikaszó hallatszik - na persze nem annyira szép, mint a vasárnapi ebédkor a tévéből, de azért maradjuk annyiban, hogy nem rossz. A zenére elkezdtünk táncolni a fű közepén, négykézlábon, mint minden normális ember tenné ebben a helyzetben. Sajnos nem volt a közelben kelta templom, de azért nélküle is egész jól elvoltunk. Persze kelta templommal táncolni mindig jobb.

A függvény feladata a nyilvánosság számára elérhetővé tenni a zenét, azaz írjon ki annyi hangjegy (U+266A) karaktert a képernyőre, egymás után, egy sorban, amilyen hosszú a zene (ezt a paraméterben érkező érték írja le). A zenét nem mással zárjuk, mint a sorköz táncoló ritmusával.

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass! A függvény inputjai a paraméterek.

```
void zene(int)
```

5. feladat: TAPSIKA (5 pont)

Mint már korábban említettem, valami fenomenális muzsikának lehettünk fültanúi. Az uwuknak is tetszett az előadás, így annak végeztével elkezdtek tapsikolni, mint azok az emberek, akik megtudták, hogy a szobát szúnyogok szállták meg. Milyen örömteli pillanatok ezek!

De vissza a produkcióra: szeretnénk megszámolni, hogy összesen hány taps hangzott el. A függvény feladata, hogy beolvasson értékeket a standard inputrol, amelyek az egyes uwuk által tapsolt mennyiséget jelzik, majd meghatározza, hogy összesen hány tapsolás történt.

A számsorozat végét a -1-es érték jelzi, ezt természetesen nem kell beszámítani.

Példa input: 6 2 4 -1

Példa output: 12

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 tapsika()
```

6. feladat: FŰ (5 pont)

Az uwuk a világszínvonalú előadás után megéheztek, így a korábban összegyűjtött fűszálak köré gyűltek, hogy elfogyasszák azokat. Viszont egy fontos kérdés, hogy el tudják-e osztani igazságosan. Én, mint igen komoly fű-szakértő mondom, hogy a fűszálakat csak egyben lehet megenni, tehát egy adott fűszálat muszáj egy uwunak elfogyasztania, nem tudják elosztani többen. Persze többet is ehetnek ha éhesek, csak arra kell vigyázni, hogy ne vigyék túlzásba, mert az ki tudja milyen mellékhatásokkal járhat (én természetesen nem).

A függvény paraméterben várja, hogy hány fűszál van összesen, illetve hogy hány uwu van, akik szeretnék egymás között elosztani a fűszálakat. A kérdés, hogy el tudják-e osztani úgy, hogy mindenkinek ugyanannyi fűszal jusson. A függvény térjen vissza 1-gyel, ha el tudják igazságosan osztani, 0-val különben.

1. PÉLDA

input: 12, 3

output: 1
magyarázat: 12 fűszál van es 3 uwu. Mindegyik 4-et eszik, így el tudják osztani
2. PÉLDA
input: 14, 5
output: 0
magyarázat: 14 fűszál van es 5 uwu. Ha mindegyik uwu eszik 2-t, akkor kimarad 4 fűszál.

Azonban 3-at már nem tud mindenki enni, mert valakinek nem jut (15 kéne), tehát nem tudták igazságosan elosztani.

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 igazsagos(int, int)
```

7. feladat: CSODÁLATOS ÁLLAT (6 pont)

Az uwuk nagyon sokáig lakomáztak, így kénytelenek voltunk továbbállni... akarom mondani továbbmászni, hogy újabb érdekességeket fedezzünk fel a varázslatos kertünkben. Nem kellett sokat másznunk, mikor újabb érdekességre lettünk figyelmesek: egy nagyon különleges, gyönyörűszép állatra. Ez egy szürke színű állat volt, hatalmas volt a feje, volt két gyönyörűséges füle, zöld szemei és hatalmas rémisztő fogai. Aztán meg nyávogott egyet. Szerencsére senki nem erodálta meg, ez nekünk sem célunk, olyan gaztettre sosem lennénk képesek - bár azért hozzáteszem a gazokat érdemes lenne a kertből kiírtani, mert a cseresznyefákra már nem is tudnak a százlábúak felmászni, de ez most mellékes a küldetésünk szempontjából.

Visszatérve a cicára. Ez nem egy hagyományos cica volt, hanem egy tündércica, aki különleges kódnyelven beszélt - illetve nyávogott. Ebben a kódnyelvben - mint az közismert - a hangok ismétlődését kell figyelnünk, mert ez alapján dekódolhatjük az üzenetet. A tündércica által mondott szöveg egy adott karakterrel kezdődik, és azt kell megfigyelnünk, hogy ez a karakter mikor fordul elő legközelebb a szövegben, és a visszafejtett kód az ennyiedik betű lesz az ABC-ben. Ezután folytatjuk a következő karakterrel. Pl. "meoowmuwu" - itt az első betű az 'm' betű, tehát a következő 'm' betűt kell megkeresnünk, ami ettől 5 karakternyi távolságra van, tehát ez egy E betű. Az ezt követő betű az 'u', tehát a következő 'u' betűt keressük, ami a szövegben 2 karakternyi távolságra található, tehát ez egy B betű, mert az az ABC második betűje. Azaz a tündércicuska által mondott szó: EB. Valószínűleg szegény nagyon megijedt egy kutyától, de szerencsére van szárnya, így el tudott repülni: huss.

A függvény feladata, hogy a standard inputról olvasson be karaktereket (egészen addig, amíg sortörést nem kap), dekódolja a szöveget, majd pedig kiírja a standard outputra a dekódolt szöveget csupa nagy betűkkel. A kiíratás természetesen a tündéri üzenet végét jelző sorszakadás varázslatos rezgésével zárul.

Tipp: garantálható, hogy az inputban mindegyik karakternek megvan a párja, és a pár záró tagja után közvetlenül érkezik a sortörés.

Példa input: akmbkrklckvviakmbjbkqkvllckoiuxjyvkqikkkkkkkkkkbjdjfjezbkwmi

Példa output: MEOW

magyarázat: először az 'a' betű párját keressük, ami 13 karakter távolságra van (M), utana a 'k' betű párját, ami 5 karakter távolságra (E), utána a 'q' betű párját, ami 15 karakter távolságra (O), végül az 'i' betű párját, ami 23 karakter távolságra (W).

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass!

```
void gyonyoruszep_allat_titkos_kodnyelvenek_visszafejtese()
```

8. feladat: BÚCSÚ (1 pont)

Gratulálok, szép munka. Már csak egy feladat vár rád. Ne várasd meg, mert az nem kelt jó első benyomást. Már csak egy egyszerű kérdésre kell válaszolni: melyik az a gyümölcs, amelyik égig ér a kertünkben?

A függvény feladata, hogy kiírja a képernyőre ennek a gyümölcsnek a nevét csupa nagybetűvel, végül pedig a sortördelés ősi rituáléjával zárjuk világszínvonalú munkánkat.

Kódold le a függvényt C nyelven! A függvény fejlécén ne változtass!

```
void noveny()
```