Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2012. május 19.**

1. Írjon **logikai függvényt**, amely paraméterként megkap egy **karaktereket** tartalmazó 3×3-as **mátrixot**, és igazat ad vissza, ha a mátrix a **tic-tac-toe** játék egy **érvényes állását** tartalmazza, különben pedig hamisat! Az állás csak akkor érvényes, ha az alábbi feltételek mindegyike teljesül:
   - a mátrix minden eleme a szóköz, az 'X' vagy az 'O' karakterek egyike;
   - az 'X'-ek száma vagy megegyezik az 'O'-k számával, vagy eggyel nagyobb nála;
   - nincs mindkét játékosnak hármasa (egy sorban, egy oszlopban vagy egy átlóban).
2. Írjon **függvényt**, amely paraméterként megkapja egy olyan **állomány nevét**, amelyben soronként egy **tantárgy neve** és egy egész **érdemjegy** szerepel, egymástól **egy vesszővel** elválasztva, és **visszaadja** az állományban található **érdemjegyek átlagát**! A tantárgyak nevében nincs vessző, viszont minden más karakter (akár szóköz is) szerepelhet benne. Az állományban egyik sor sem hosszabb 40 karakternél. Figyeljen arra, hogy egész számok átlaga valós!
3. Írjon **logikai függvényt**, amely paraméterként megkap **egy sztringet** és **két karaktert**, és igaz értékkel tér vissza, ha a sztringben **mindkét karakter legalább kétszer** előfordul, különben pedig hamissal!
4. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, **egészeket** tartalmazó kétdimenziós **tömböt**, és előállít egy **vektort**, amely a megkapott tömb azon elemeit tartalmazza, amelyek **indexeinek összege páros**! Az előállított vektort a hívónak is látnia kell!
5. Írjon **programot**, amely a billentyűzetről valós számokat olvas be a bemenet végéig, és egy **szöveges állományban** elhelyezi a beolvasott számok közül azokat, amelyeknek a tört része (abszolút értékben) nagyobb, mint 0,5! Az állomány nevét a program az első parancssori argumentumában kapja meg.

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2012. május 25.**

Adott az alábbi struktúra:

```
struct aru
{
  char *nev;
  int ar;
};
```

1. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, `struct aru` típusú **árukat** tartalmazó egydimenziós **tömböt**, és visszatér a **legdrágább** áru **nevével**!
2. Írjon konverziós **függvényt**, amely paraméterként megkap egy **sztringet**, amely egy **áru** adatait tartalmazza úgy, hogy a sztring utolsó szóközéig az áru **neve**, utána pedig az **ára** szerepel (az utolsó szóköz csak elválasztó szerepkörű, nem képezi részét az áru nevének)! A függvény adja vissza az árut egy `struct aru` típusú **struktúrában**!
3. Írjon konverziós **függvényt**, amely paraméterként megkap egy `struct aru` típusú **árut**, és visszaad egy **sztringet**, amely a paraméterként megkapott árut reprezentálja a 2. feladatban megadott formátumban! (Segítség: egy egész szám sztringgé konvertálásához lásd az `sprintf()` függvényt!)
4. Írjon **programot**, amely az `aruk.txt` nevű szöveges **állományból** soronként **árukat** olvas be, és **képernyőre** írja az összes olyan áru **nevét**, amelynek az **ára** egy **parancssori argumentumként** megadott egész szám! Az állomány sorai a 2. feladatban megadott formátumúak, és egyik sem hosszabb 100 karakternél.
5. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, **áruneveket** tartalmazó kétdimenziós **tömböt**, meghatározza a **leghosszabb árunevet** tartalmazó **oszlopot**, majd előállít egy `struct aru` típusú elemeket tároló **vektort** az ebben az oszlopban szereplő árukból! A vektorban az egyes áruk **ára** a **nevük hossza** legyen! **A vektort a hívónak is látnia kell!**

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2012. május 26.**

1. Írjon **programot**, amely a billentyűzetről egész számokat olvas be, amíg 0-t nem kap, majd kiírja a képernyőre a beolvasott **páratlan** és **páros** számok **darabszámát**! A sorozat végét jelző 0 nem képezi részét a sorozatnak.
2. Írjon **programot**, amely a `film.sub` nevű, filmfeliratot tartalmazó **szöveges állományban** megkeresi a **legrövidebb** ideig kiírt feliratsor(oka)t! Az állomány sorai a következő formátumúak:

   `{kezdő}{vég}szöveg`

   A `kezdő` és a `vég` nemnegatív egész számok. Azon sor(ok) `szöveg` részét kell tehát a képernyőre írni, amely(ek)ben a `vég–kezdő` minimális.
3. Írjon **eljárást**, amely egy paraméterként megkapott **sztringben** minden **második szóköz** karaktert **csillag** karakterre (*) cseréli!
4. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, **karaktereket** tartalmazó kétdimenziós **tömböt**, és visszaadja annak az **oszlopnak** az **indexét**, amelyik a **legtöbb** angol **magánhangzót** (a, e, i, o, u, A, E, I, O, U) tartalmazza! Ha több ilyen oszlop is van, akkor a **legutolsó** ilyen oszlop indexét kell visszaadni.
5. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, **valósakat** tartalmazó egydimenziós **tömböt**, és visszaad egy **vektort**, amely a megkapott tömb elemeit tartalmazza **sztringként**! (Segítség: a konverzióhoz lásd az `sprintf()` függvényt!)

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2013. május 28.**

1. Írjon **programot**, amely az első parancssori argumentumában megadott **állomány** azon **sorait**, amelyek **hosszabbak** a második parancssori argumentumban megadott egész számnál, a harmadik parancssori argumentumban megadott **állományba másolja**! Ha valamelyik állomány nem nyitható meg, a program írjon ki egy hibaüzenetet, és álljon meg. Felteheti, hogy a bemeneti állomány sorai nem hosszabbak 100 karakternél.
2. Írjon **függvényt**, amely paraméterként megkap egy pozitív egész számot, és visszaadja a szám **különböző prímosztóinak** a számát!
3. Írjon **eljárást**, amely a paramétereként megkapott sztring minden **mondatát nagy kezdőbetűssé** alakítja! A mondatok első karakterének a mondatzáró írásjeleket (pont, kérdőjel, felkiáltójel) követő első nem fehér karaktert, valamint a sztring első nem fehér karakterét tekintjük.
4. Írjon **logikai függvényt**, amely igazat ad vissza, ha a paramétereként megkapott három valós szám reprezentálhatja egy **derékszögű háromszög oldalhosszait**!
5. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, nemnegatív **egészeket** tartalmazó **négyzetes mátrixot**, és előállít egy vektort, amely a mátrix **mellékátlójában** található elemek **számjegyeinek a számát** tartalmazza a jobb felső elemtől a bal alsó elemig haladva! Az előállított vektort a hívónak is látnia kell!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2013. május 31.**

1. Írjon **programot**, amely az első parancssori argumentumában egy **szöveges állomány** nevét kapja meg, amelynek minden sorában egy legfeljebb 100 karakter hosszúságú URL (sztring) szerepel! A program írja képernyőre a **legtöbb pont** karaktert tartalmazó URL-t (ha több ilyen is van, akkor az utolsót)! Ha nincs megadva parancssori argumentum, vagy az állomány nem nyitható meg, a program írjon ki egy hibaüzenetet.
2. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, URL-eket tartalmazó **vektort** és egy sztringet, és visszaadja azoknak az **URL-eknek a számát**, amelyek legfelső szintű tartománya (top-level domain; az URL **utolsó pont utáni része**) megegyezik a kapott sztringgel! Felteheti, hogy minden URL tartalmaz legalább egy pont karaktert.
3. Írjon **eljárást**, amely a paramétereként megkapott URL-t úgy módosítja, hogy az a **ponttal elválasztott** részeit **fordított sorrendben** tartalmazza! A „www.inf.unideb.hu” URL-t például a „hu.unideb.inf.www” URL-re kell cserélnie.
4. Írjon **függvényt**, amely paraméterként megkap egy **négyelemű bájttömböt** (`unsigned char` típusú elemekkel), és visszaad egy **IP címet** sztringként, amely az egyes bájtok értékét tartalmazza egy-egy **ponttal elválasztva** (pl. 193.6.135.21)!
5. Adott az alábbi struktúra:

```
struct dns
{
  char *nev;             /* pl. www.inf.unideb.hu */
  unsigned char ip[4];   /* pl. 193.6.135.21 */
};
```

   Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, `struct dns` típusú elemeket tartalmazó **vektort**, valamint egy állománynevet, és **kiírja** a megadott nevű **állományba** soronként a vektorban található neveket és IP címeket a fenti példában megadott formátumban, egy tabulátorral elválasztva, **név szerint rendezett** sorrendben!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2013. június 1.**

1. Írjon **függvényt**, amely paraméterként megkap egy 8×8-as karaktertömbbel reprezentált **sakktáblát**, amelyben a szóköz értékű elemek az üres mezőket jelölik, a többi pedig a különböző figurákat! A függvény adja vissza egy újonnan létrehozott **egydimenziós karaktertömbben** a táblán lévő **figurákat** jelölő karaktereket (tetszőleges sorrendben)! A tömb címét és méretét a függvény egy **struktúrában** adja vissza!
2. Írjon **eljárást**, amely paraméterként megkap egy 8×8-as karaktertömbbel reprezentált **sakktáblát**, amelyben a szóköz értékű elemek az üres mezőket jelölik! Az eljárás **cserélje ki** az **üres mezőket** jelölő elemeket egy **v** betűre, ha az a mező **világos**, illetve **s** betűre, ha **sötét**! A tábla pepita, és a (0,0) pozíció (a tábla bal felső sarka) világos.
3. Írjon **programot**, amely beolvas a billentyűzetről egy *n* egész számot, és képernyőre írja, hogy **hány** olyan **sora** van az első parancssori argumentumban megadott nevű állománynak, amelyben **minden szó** legalább *n* karakter hosszúságú! Szónak tekintjük az egymástól szóközcsoportokkal elválasztott karaktersorozatokat. Felteheti, hogy az állomány sorai legfeljebb 100 karakter hosszúak. Ha a megadott nevű állomány nem nyitható meg, a program írjon ki egy hibaüzenetet, és álljon meg.
4. Írjon **logikai függvényt**, amely paraméterként megkap egy tetszőleges méretű, **valósakat** tartalmazó mátrixot, és igazat ad vissza, ha a mátrix **minden oszlopában több nemnegatív** érték szerepel, mint negatív!
5. Írjon **eljárást**, amely paraméterként megkap egy **sztringet**, amelyet lerövidít úgy, hogy az egymás mellett álló **azonos karakterekből** csak **egyet** hagy meg! Az „aaaalmmmaa” sztringből például az „alma” sztringet kell előállítania.

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2014. május 30.**

1. Írjon **függvényt**, amely paraméterként megkap egy **sztringet**, amely a hét egy időpontját tartalmazza a következő formában:

   napnév_rövidítés óra.perc

   A *napnév_rövidítés* a következők valamelyike: „H”, „K”, „Sze”, „Cs”, „P”, „Szo”, „V”. Az időpont adat 24 órás formában van megadva, azaz az *óra* 0 és 23, a *perc* 0 és 59 közötti értéket vehet fel. A függvény visszatérési értékként határozza meg, hogy **hány perc telt el** a hétből a megadott időpontig, ha feltételezzük, hogy a hét hétfőn 0.00-kor kezdődik, és nem esik a hétre tavaszi vagy őszi óraátállítás! (Segítség: egy sztring egész számmá konvertálásához lásd az `atoi()` vagy az `sscanf()` függvényt!)

2. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, **valósakat** tartalmazó **mátrixot**, és visszaad egy új, valósakat tartalmazó egydimenziós tömböt, amely az eredeti tömb **oszlopaiban** található **számok átlagát** tartalmazza!

3. Írjon **programot**, amely az első parancssori argumentumaként megadott szöveges **állományból** az alábbi szerkezetű sorokat olvassa állományvégjelig:

   településnév: *hőmérséklet*,*hőmérséklet*[,*hőmérséklet*]...

   A településnév egy legfeljebb 30 karakter hosszú sztring, a hőmérsékleti adatok előjeles egész számok, a teljes sor hossza pedig nem haladja meg az 1000 karaktert. Egy-egy sor az adott településre vonatkozóan egy adott időintervallumban mért hőmérsékleti adatokat tartalmaz. A mérések pontos darabszámát nem ismerjük, azt azonban tudjuk, hogy az időintervallum során legalább két, de legfeljebb ötszáz mérést végeznek. A program minden település esetén **írja a standard kimenetre** az időintervallumra vonatkozó **hőingadozást**, azaz a legnagyobb és legkisebb mért hőmérséklet különbségét! A kimenet formátuma soronként az alábbi legyen:

   településnév: *hőingadozás*

   (Segítség: egy sztring egész számmá konvertálásához lásd az `atoi()` vagy az `sscanf()` függvényt!)

4. Írjon **programot**, amely a standard bemenetről soronként **egész számokat olvas be**! Minden sorban az első szám azt mondja meg, hogy hány további szám található még az adott sorban. A program minden sor esetén a második számtól kezdődően számolja össze a pozitív és a negatív számokat, majd **írja a standard kimenetre** azoknak a soroknak a **sorszámát**, amelyekben **több pozitív számot** talált, mint negatívat! A bemenetet egy olyan sor zárja, amely csak egyetlen 0-t tartalmaz, ezt a sort a programnak már nem kell feldolgoznia.

5. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, sztringeket tartalmazó **vektort**, valamint egy állománynevet, és **kiírja** a megadott nevű **állományba** soronként a vektorban található sztringeket a **hosszuk szerinti növekvő** sorrendben!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2014. május 31.**

1. Írjon **függvényt**, amely paraméterként megkap **két nem üres sztringet**, és visszatérési értékként meghatározza, hogy – az esetleges átfedéseket is figyelembe véve – **hányszor fordul elő** az első sztringben a második!
2. Írjon **függvényt**, amely paraméterként megkap egy nem üres **sztringet**, és visszaadja annak **leghosszabb borderét** egy új sztringként!
3. Írjon **programot**, amely az első parancssori argumentumaként megadott szöveges **állományból** soronként könyvek adatait olvassa be állományvégjelig! Egy-egy sor a következő alakú:

   szerző[,*szerző*]...;*könyvcím*;*oldalszám*

   Ez a leírás azt jelenti, hogy egy könyvnek több szerzője is lehet, de legalább egy mindegyik könyvnek van. A szerzők nevei egyenként legfeljebb 30, a könyv címe legfeljebb 100 karakter hosszúságú. A sor teljes hossza nem haladja meg az 1000 karaktert. A könyvek oldalszáma minden esetben egy pozitív egész szám.

   A program határozza meg és írja a standard kimenetre a **legtöbb oldalon kinyomtatott könyv címét**! Ha több azonos oldalszámú könyv is megfelel ennek a kritériumnak, akkor közülük a felsorolás szerinti legelsőnek a címét kell a programnak kiírnia.

   (Segítség: egy sztring egész számmá konvertálásához lásd az `atoi()` vagy az `sscanf()` függvényt!)
4. Írjon **programot**, amely a standard bemenetről **egész számokat olvas be** a 0 végjelig! A program határozza meg és írja a standard kimenetre azt a **három** (0-tól különböző) értéket, amelyek **átlaga maximális**! Ha a program háromnál kevesebb 0-tól különböző értéket olvas, akkor írja ki mindet!
5. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, sztringeket tartalmazó **vektort**, valamint egy állománynevet, és **kiírja** a megadott nevű **állományba** soronként a vektorban található sztringeket a **bennük lévő szóközök száma szerinti növekvő** sorrendben!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2014. június 2.**

Adott az alábbi struktúra:

```
struct kocsi
{
  char *marka, *tipus;
  int loero;
};
```

1. Írjon **eljárást**, amely paraméterként megkap egy tetszőleges méretű, `struct kocsi` típusú **autókat** tartalmazó egydimenziós **tömböt**, és a standard kimenetre írja a **legerősebb** autók **márkáját és típusát**!
2. Írjon konverziós **függvényt**, amely paraméterként megkap egy **sztringet**, amely egy **autó** adatait tartalmazza úgy, hogy a márkát a típustól, valamint a típust a lóerőtől egy-egy pontosvessző választja el! A függvény adja vissza az autót egy `struct kocsi` típusú **struktúrában**! (Segítség: egy sztring egész számmá konvertálásához lásd az `atoi()` vagy az `sscanf()` függvényt!)
3. Írjon konverziós **függvényt**, amely paraméterként megkap egy `struct kocsi` típusú **autót**, és visszaad egy **sztringet**, amely a paraméterként megkapott autót reprezentálja a 2. feladatban megadott formátumban! (Segítség: egy egész szám sztringgé konvertálásához lásd az `sprintf()` függvényt!)
4. Írjon **programot**, amely az `autok.txt` nevű szöveges **állományból** soronként **autókat** olvas be, és a standard kimenetre írja, hogy **hány** olyan autó szerepel az állományban, amelynek a **teljesítménye nagyobb** az autók **átlagos teljesítményénél**, és hogy ez **hány százaléka a teljes darabszámnak**! Az állomány sorai a 2. feladatban megadott formátumúak, és egyik sem hosszabb 100 karakternél.
5. Írjon **függvényt**, amely paraméterként megkap egy tetszőleges méretű, `struct kocsi` típusú **autókat** tartalmazó kétdimenziós **tömböt**, meghatározza a **leghosszabb márkanevű** autót tartalmazó **oszlopot**, majd előállít és visszaad egy sztringeket tároló **vektort**, amely az ebben az oszlopban szereplő autók márkáját és típusát tartalmazza egy szóközzel elválasztva!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2015. május 29.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **logikai függvényt**, amely a paramétereként megkapott tetszőleges hosszúságú **sztringről** eldönti, hogy az a **C/C++ nyelvek azonosítókra** vonatkozó elnevezési konvencióinak megfelel-e! Ha igen, akkor a függvény igaz értéket, ha nem, hamisat adjon vissza! A C/C++ nyelvekben az elnevezési konvenciók szerint az azonosítókban csak kisbetűket használnak, a többszavas azonosítókban pedig a szavakat az aláhúzás ('_') karakterrel választják el egymástól. Példa C/C++ azonosítókra: `i`, `long_and_mnemonic_identifier`.
2. Írjon **programot**, amely megvizsgálja, hogy **parancssori argumentumai** mindannyian **párosával** fordulnak-e elő! Ha minden parancssori argumentum páros sokszor fordul elő, akkor a program a „YES”, egyébként a „NO” sztringet írja a standard kimenetre!
3. Írjon **függvényt**, amely paraméterként megkapja egy **sztringeket tartalmazó kétdimenziós tömb** kezdőcímét, sorainak és oszlopainak számát, valamint egy sztringet! A függvény az eredeti tömb módosítása nélkül hozzon létre egy **új, egészeket tartalmazó egydimenziós tömböt**, amelynek annyi eleme van, ahány oszlopa az eredeti tömbnek volt! A függvény az új tömböt attól függően töltse föl logikailag igaz (1) vagy hamis (0) értékekkel, hogy a paraméterként megkapott **sztring megtalálható-e** az eredeti kétdimenziós tömb adott **oszlopában**, majd adja vissza az új tömb kezdőcímét!
4. Írjon **programot**, amely az első parancssori argumentumaként megadott **szöveges állományból** soronként **időpontok egy listáját** olvassa be! Az időpontok HH.MM alakúak, ahol HH egy 0 és 23, MM pedig egy 0 és 59 közötti egész szám, utóbbi 10-nél kisebb értékek esetén egy vezető 0-val kiegészítve. Az időpontokat egymástól pontosvessző karakterek választják el. A sorok hossza nem haladja meg a 200 karaktert. A program írja a standard kimenetre soronként az egyes sorokban található **legkésőbbi (legnagyobb) időpontot**!
5. Írjon **programot**, amely a standard bemenetről soronként **egész számokat** olvas be, és meghatározza közülük a **legkisebbnek** és a **legnagyobbnak** az értékét! A bemenet első sora pontosan egy darab pozitív egész számot (N) tartalmaz, ami a feldolgozandó tesztesetek számát jelzi. A következő N sor mindegyikében egész számok szerepelnek. Minden sor első száma (K, ahol $1 \leq K \leq 10$) azt árulja el, hogy hány további szám található a sorban. Programjának ezek közül a számok közül kell kiválasztania a legkisebbet és a legnagyobbat, majd ezt a két értéket pontosan egy szóköz karakterrel elválasztva a standard kimenetre írnia minden feldolgozott teszteset esetén!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2015. június 5.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **eljárást**, amely paraméterként megkap egy **állománynevet**, egy **sztringeket tartalmazó egydimenziós tömb** kezdőcímét, valamint a tömb elemszámát! Az eljárás **a tömb módosítása nélkül** írja ki a megadott nevű szöveges állományba soronként a tömb azon elemeit **megfordítva**, amelyek **nem angol betűvel kezdődnek**! (Segítség: lásd az `isalpha()` függvényt.)
2. Írjon **programot**, amely a „YES” sztringet írja a standard kimenetre, ha **minden parancssori argumentuma csak azonos karaktereket** tartalmaz (külön-külön), egyébként pedig a „NO” sztringet!
3. Írjon **függvényt**, amely paraméterként megkapja egy dupla pontosságú **valósakat tartalmazó kétdimenziós tömb** kezdőcímét, valamint sorainak és oszlopainak számát! A függvény az eredeti tömb módosítása nélkül hozzon létre egy **új**, dupla pontosságú **valósakat tartalmazó egydimenziós tömböt**, amelynek annyi eleme van, ahány sora az eredeti tömbnek volt! A függvény az új tömböt töltse fel az eredeti tömb egyes **soraiban található elemek összegeinek gördülő összegével**, majd adja vissza az új tömb kezdőcímét! Másképpen fogalmazva: az új tömb *i*-edik eleme az eredeti tömb első *i* sorában található elemek összegét tartalmazza!
4. Írjon **programot**, amely az első parancssori argumentumaként megadott **szöveges állományból** soronként **repülőgéptípusok egy listáját** olvassa be! Egy-egy sor a következő alakú:

   gyártó;*típus*;*hossz*

   A *gyártó* és a *típus* szóközt és pontosvessző karaktert nem tartalmazó, legfeljebb 50 karakter hosszúságú sztring, a *hossz* pedig egy valós szám legfeljebb 8 karakteren. A program írja a **standard kimenetre** soronként azoknak a repülőgépeknek a gyártóját és típusát egy szóközzel elválasztva, amelyeknek a **hossza legalább** a második parancssori argumentumban megadott érték! (Segítség: egy sztring valós számmá történő konvertálásához lásd az `atof()` vagy az `sscanf()` függvényeket.)
5. Írjon **programot**, amely a **standard bemenetről** soronként legfeljebb 100 karakter hosszúságú **sztringeket** olvas be, majd a **standard kimenetre** írja közülük soronként azokat, amelyek **az átlagosnál hosszabbak**! A kimenetet **rendezze** a **sztringek hossza szerint csökkenő** sorrendbe! Ha két vagy több sztring is azonos hosszúságú lenne, akkor őket rendezze **lexikografikusan növekvő** sorrendbe! Felteheti, hogy a bemenet legfeljebb 100 sort tartalmaz.

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2015. június 13.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

Adott az alábbi struktúra:

```
struct orszag
{
  char *nev;
  int terulet, nepesseg;
};
```

1. Írjon konverziós **függvényt**, amely paraméterként megkap egy `struct orszag` típusú struktúrát, és visszaadja annak a **sztring megfelelőjét**! A sztring elején az ország neve, majd egy szóközt követően zárójelben a területe és a népessége szerepeljen egy vesszővel elválasztva! (Segítség: egy sztringbe történő formázott íráshoz lásd az `sprintf()` függvényt.)
2. Írjon **eljárást**, amely paraméterként megkapja egy `struct orszag` típusú elemeket tartalmazó **egydimenziós tömb** kezdőcímét, a tömb elemszámát, valamint egy állománynevet, és **kiírja** a megadott nevű **állományba** soronként, a tömbbeli előfordulásuk sorrendjében **az átlagosnál nagyobb területű** országok **nevét és népességét** egy szóközzel elválasztva!
3. Írjon **programot**, amely a `bemenet.txt` nevű **szöveges állományból** soronként országok adatait olvassa be az állomány végéig! Egy-egy sor felépítése a következő:

   név;*terület*;*népesség*[;*népesség*]…

   A *név* egy legfeljebb 30 karakteres, pontosvesszőt nem tartalmazó sztring, a *terület* és a *népesség* pozitív egész számok. Az ország teljes népességét a sorban található népességértékek összege adja meg. Az állomány legfeljebb 200 sorból áll, amelyek legfeljebb 1000 karakter hosszúak. A program írja **a standard kimenetre** soronként azoknak az országoknak a nevét, területét és teljes népességét egy-egy vesszővel elválasztva, amelyek **népsűrűsége** (a népesség és a terület hányadosa) **meghaladja** a program **első parancssori argumentumában** megadott valós értéket! A kimenetet **rendezze népsűrűség szerint csökkenő**, azonos népsűrűségek esetén **országnév szerint növekvő** sorrendbe! (Segítség: egy sztring egész, illetve valós számmá történő konvertálásához lásd az `atoi()`, az `atof()` vagy az `sscanf()` függvényeket, egy sztring feldarabolásához az `strtok()`, a rendezéshez pedig a `qsort()` függvényt.)
4. Írjon **függvényt**, amely paraméterként megkapja egy `struct orszag` típusú elemeket tartalmazó **kétdimenziós tömb** kezdőcímét, sorainak és oszlopainak számát, valamint egy dupla pontosságú valós számot (*s*)! A függvény az eredeti tömb megváltoztatása nélkül hozzon létre egy **új egydimenziós tömböt**, amely azoknak az **országoknak a nevét** tartalmazza csupa **nagybetűssé** alakítva, amelyek **népsűrűsége** (a népesség és a terület hányadosa) *s* – 10 és *s* + 10 **közé esik**! Az országnevek az eredeti tömb **sorfolytonos** bejárásának sorrendjében szerepeljenek az új tömbben! A tömb **utolsó elemében NULL** pointer szerepeljen, ezzel jelezve a tömb végét! A függvény visszatérési értéke az új tömb kezdőcíme legyen! (Segítség: egy karakter nagybetűssé történő konvertálásához lásd a `toupper()` függvényt.)
5. Írjon **programot**, amely a **standard bemenetről** soronként országok adatait olvassa be állományvégjelig! Egy-egy sor felépítése a következő:

   név;*terület*;*népesség*[;*népesség*]…

   A *név* egy legfeljebb 30 karakteres, pontosvesszőt nem tartalmazó sztring, a *terület* és a *népesség* pozitív egész számok. Az ország teljes népességét a sorban található népességértékek összege adja meg. A program írja **a standard kimenetre** a **legnagyobb területű országnak a nevét**! Ha több ilyen ország is létezne, akkor a bemeneten leghamarabb előforduló ország nevét kell kiírni. Figyeljen arra, hogy a sorok tetszőleges hosszúságúak lehetnek! (Segítség: egy sztring egész számmá történő konvertálásához lásd az `atoi()` vagy az `sscanf()` függvényeket, egy sztring feldarabolásához pedig az `strtok()` függvényt.)

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2016. május 25.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **programot**, amely a **standard bemenetről soronként** egy vagy több nemnegatív egész számot olvas! Ha a sor elsőként beolvasott száma 0, a program fejezze be a működését. Ha a sor első száma pozitív, akkor az azt jelzi, hogy a sorban hány további szám található. A program írja a „YES” karakterláncot a **standard kimenetre**, ha **a sor második száma megegyezik a sor utolsó számával**, egyébként pedig a „NO” karakterláncot írja a kimenetre!
2. Írjon **függvényt**, amely **paraméterként** megkap egy **sztringet**, amely a **hét egy időpontját** tartalmazza a következő formában:

   napnév_rövidítés óra.perc

   A *napnév_rövidítés* a következők valamelyike: „H”, „K”, „Sze”, „Cs”, „P”, „Szo”, „V”. Az időpont adat 24 órás formában van megadva, azaz az óra 0 és 23, a perc 0 és 59 közötti értéket vehet fel. A függvény visszatérési értékként határozza meg, hogy **hány perc telt el a hétből a megadott időpontig**, ha feltételezzük, hogy a hét hétfőn 0.00-kor kezdődik, és nem esik a hétre tavaszi vagy őszi óraátállítás!
3. Írjon **programot**, amely a **standard kimenetre** írja egy **adott szak** hallgatói közül a **három legjobb tanulmányi eredménnyel** rendelkezőnek az adatait, vagy az összesét, ha az adott szakon nincs több hallgató háromnál! A hallgatók adatai **tanulmányi átlag szerint csökkenő sorrendben**, azonos tanulmányi átlagok esetén pedig **Neptun-kód szerinti ábécérendben** jelenjenek meg a standard kimeneten! A tanulmányi átlagokat **egy tizedesjegy** pontossággal kell kiírni.

   A feldolgozandó adatokat a program a **standard bemenetről** olvassa. A bemenet **első sorában egy szak** legfeljebb 10 karakter hosszú rövidítése fog szerepelni. A bemenet további, legfeljebb 1000 sorának alakja a következő:

   vezetéknév keresztnév neptun_kód szak tanulmányi_átlag

   A *vezetéknév* és a *keresztnév* egy-egy legfeljebb 30 karakter hosszúságú sztring, a *neptun_kód* pontosan 6 karakterből áll, a *szak* egy legfeljebb 10 karakter hosszúságú rövidítés, a *tanulmányi_átlag* pedig egy valós szám. A sztringek egyike sem tartalmaz szóköz karaktert.
4. Írjon **programot**, amely az első **parancssori argumentumaként** megadott **szöveges állományból** az alábbi szerkezetű sorokat olvassa **állományvégjelig**:

   településnév: *hőmérséklet*,*hőmérséklet*[,*hőmérséklet*]…

   A *településnév* egy legfeljebb 30 karakter hosszú sztring, a hőmérsékleti adatok előjeles egész számok, a teljes sor hossza pedig nem haladja meg az 1000 karaktert. Egy-egy sor az adott településre vonatkozóan egy adott időintervallumban mért hőmérsékleti adatokat tartalmaz. A mérések pontos darabszámát nem ismerjük, azt azonban tudjuk, hogy az időintervallum során legalább két, de legfeljebb ötszáz mérést végeznek. A program minden település esetén írja a **standard kimenetre** az időintervallumra vonatkozó **hőingadozást**, azaz a **legnagyobb és legkisebb mért hőmérséklet különbségét**! A kimenet formátuma soronként az alábbi legyen:

   településnév: *hőingadozás*
5. Írjon **függvényt**, amely **paraméterként** megkapja egy **dupla pontosságú valósakat tartalmazó kétdimenziós tömb** kezdőcímét, valamint sorainak és oszlopainak számát! A függvény az eredeti tömb módosítása nélkül hozzon létre egy **új**, dupla pontosságú valósakat tartalmazó **egydimenziós tömböt**, amely az eredeti tömb **oszlopaiban található számok átlagát** tartalmazza, és **adja vissza e tömb kezdőcímét**!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2016. május 27.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **programot**, amely a **standard bemenetről** angol szavakat tartalmazó, legfeljebb 50 karakter hosszúságú **sorokat olvas** az első **üres sorig**! A szavak kizárólag az angol ábécé betűit tartalmazzák, és egy sorban két egymást követő szót pontosan egy szóköz karakter választ el egymástól. Az első szó előtt és az utolsó szó után nincsenek szóközök. A program minden sor esetén írja a **standard kimenetre**, hogy **hány szó szerepel az adott sorban**!

2. Írjon **függvényt**, amely **paraméterként** megkap egy **sztringet**, és visszatérési értékként meghatároz egy egész számot! A sztring egy áramerősség-értéket tartalmaz a következő formában:

   előjel_nélküli_egész_szám {`A`|`kA`|`mA`}

   Ebben az alakban „`A`” az amper, „`kA`” a kiloamper (az amper ezerszerese), „`mA`” pedig a milliamper (az amper ezredrésze) mértékegységet jelöli. Az *előjel_nélküli_egész_szám* egy decimális számjegyekből álló karaktersorozat. A függvény **visszatérési értéke a kapott mennyiség amperben** kiszámított értéke legyen, feltételezve azt, hogy a milliamperben megadott értékek maradék nélkül oszthatók 1000-rel!

3. Írjon **programot**, amely egy adott napon adásba kerülő **tévéműsorokat** írja a **standard kimenetre időrendi sorrendben**! Ha két műsor azonos időben kezdődne, akkor őket a program az őket vetítő **csatornák neveinek ábécérendje** szerinti sorrendben írja ki!

   A feldolgozandó adatokat a program a **standard bemenetről** olvassa. A bemenet **első sorában a hét egyik napja** fog szerepelni, az ezen a napon adásba kerülő műsorokat kell kiírnia a programnak. A bemenet további, legfeljebb 400 sorának alakja a következő:

   műsor_neve adó_neve nap óra:*perc*

   A *műsor_neve* és az *adó_neve* egy-egy legfeljebb 30 karakter hosszúságú, a *nap* egy legfeljebb 10 karakter hosszúságú sztring, míg az *óra* és a *perc* egy 0 és 24 óra közötti időpontot határoz meg. A sztringek egyike sem tartalmaz szóköz karaktert, bennük a szóközöket aláhúzás karakterek helyettesítik.

4. Írjon **programot**, amely az első **parancssori argumentumaként** megadott szöveges **állományból** egy legalább egy zeneszámot tartalmazó **zenei CD tartalomjegyzékét** olvassa be! A tartalomjegyzék a következő alakú sorokból áll:

   sorszám;*előadó*;*zeneszám_címe*;*időtartam*

   Minden tartalomjegyzék esetén a *sorszám* egy 1-ről induló, egyesével növekvő számsorozat, az *előadó* egy legfeljebb 30, a *zeneszám_címe* egy legfeljebb 100 karakter hosszúságú sztring, míg az *időtartam PP*:*MM* alakban van megadva, ahol *PP* egy nemnegatív egész szám, *MM* pedig egy 0 és 59 közötti egész érték, ahogyan az a példa bemenetben is látható.

   A program határozza meg és írja a **standard kimenetre** a **leghosszabb időtartamú zeneszámnak az előadóját**! Ha több azonos időtartamú zeneszám is megfelelne ennek a kritériumnak, akkor közülük a CD-n előrébb helyett foglaló, legkisebb sorszámú zeneszám előadóját kell a programnak a kimenetre írnia.

5. Írjon **függvényt**, amely paraméterként megkapja egy **sztringeket tartalmazó tömb** kezdőcímét és elemszámát! A függvény határozza meg a tömb **legrövidebb sztringjének** a hosszát, számolja meg, hogy hány ilyen hosszúságú sztring van a tömbben (*n* darab), majd **hozzon létre** egy **új**, *n* + 1 elemű **tömböt**, amelynek első *n* eleme az az ***n* darab mutató** legyen az eredeti tömbbeli elhelyezkedésük sorrendjében, amelyek **az eredeti tömb legrövidebb sztringjeit** címzik! Az új tömb **(*n* + 1)-edik eleme egy NULL mutató** legyen! A függvény **visszatérési értéke** az újonnan létrehozott **mutatótömb kezdőcíme** legyen!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2016. május 30.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **függvényt**, amely **paraméterként** megkap egy **sztringet** és **két** nemnegatív egész **számot**! A függvény a paramétereként megkapott sztring megváltoztatása nélkül hozzon létre és **adjon vissza** egy olyan **új sztringet**, amelyet úgy kapunk, hogy az **eredeti sztringből kivágjuk** és eltüntetjük azt a **részsztringet**, amelyik az első egész számmal jelzett pozíción kezdődik és a második számmal jelzett pozíción ér véget, az eredeti sztring megmaradt részeit pedig összefűzzük! Ha az első szám nagyobb lenne a másodiknál, illetve ha a második szám legalább akkora, mint a sztring hossza, akkor a sztringet az első szám által jelzett pozíciótól a végéig csonkolja a függvény!
2. Írjon **programot**, amely **parancssori argumentumait hosszuk szerint növekvő sorrendben** írja a standard kimenetre soronként, feltételezve, hogy minden parancssori argumentum hossza különböző!
3. Írjon **programot**, amely az első **parancssori argumentumaként** megadott szöveges **állományból** soronként **könyvek** adatait olvassa be! Egy-egy sor a következő alakú:

   szerző[,*szerző*]…;*könyvcím*;*oldalszám*

   Ez a leírás azt jelenti, hogy egy könyvnek több szerzője is lehet, de legalább egy mindegyik könyvnek van. A szerzők nevei egyenként legfeljebb 30, a könyv címe legfeljebb 100 karakter hosszúságú. A teljes sor hossza nem haladja meg az 1000 karaktert. A könyvek oldalszáma minden esetben egy pozitív egész szám.

   A program határozza meg és írja a **standard kimenetre a legtöbb oldalon kinyomtatott könyv címét**! Ha több azonos oldalszámú könyv is megfelel ennek a kritériumnak, akkor közülük a felsorolás szerinti legelsőnek a címét kell a programnak a kimenetre írnia.
4. Írjon **programot**, amely a **standard bemenetről** soronként **egész számokat** olvas be! Minden sorban az első szám azt határozza meg, hogy hány további szám található még az adott sorban. A program minden sor esetén írja a **standard kimenetre** azt, hogy a második számtól kezdve a sor végéig található számok között **hányszor fordul elő a legnagyobb értékű szám**! A bemenetet egy olyan sor zárja, amely csak egyetlen 0-t tartalmaz, ezt a sort a programnak már nem kell feldolgoznia.
5. Írjon **eljárást**, amely **paraméterként** megkap egy **állománynevet**, egy **sztringeket** tartalmazó **egydimenziós tömb** kezdőcímét, valamint a tömb elemszámát! Az eljárás a tömb módosítása nélkül **írja ki** a megadott nevű szöveges **állományba** soronként a tömb azon **elemeit megfordítva**, amelyek **nem angol betűvel kezdődnek**!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2017. május 26.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **függvényt**, amely **paraméterként** megkap egy **sztringet**! A függvény a paramétereként megkapott sztring megváltoztatása nélkül hozzon létre és **adjon vissza** egy **új sztringet**, amelyet úgy kapunk, hogy az **eredeti sztringben** megkeressük az **első pontosvessző** karaktert, és **felcseréljük** egymással a tőle **balra és jobbra** elhelyezkedő részsztringeket! Ha a megkapott sztringben egyetlen pontosvessző karakter sincs, akkor az új sztring tartalma egyezzen meg a kapott sztring tartalmával!
2. Írjon **programot**, amely a standard kimenetre írja soronként egyesével a **leghosszabb parancssori argumentumait lexikografikusan csökkenő** sorrendben!
3. Írjon **programot**, amely az első **parancssori argumentumaként** megadott szöveges **állományból** soronként **filmek** adatait olvassa be, legalább egyét! Egy-egy sor a következő alakú:

   [*szereplő*[,*szereplő*]…;]*filmcím*;*hossz*

   Ez a leírás azt jelenti, hogy egy filmnek több szereplője is lehet, de az is lehet, hogy egy sincs; utóbbi esetben a sor mindjárt a film címével kezdődik. A szereplők nevei egyenként legfeljebb 50, a film címe legfeljebb 100 karakter hosszúságú. Sem a szereplők nevei, sem a filmcímek nem tartalmaznak sem vessző, sem pontosvessző karaktert. A teljes sor hossza nem haladja meg az 1000 karaktert. A filmek (percben mért) hossza minden esetben egy pozitív egész szám.

   A program határozza meg és írja a **standard kimenetre a leghosszabb film címét**! Ha több azonos hosszúságú film is megfelel ennek a kritériumnak, akkor közülük a felsorolás szerinti legelsőnek a címét kell a programnak a kimenetre írnia.
4. Írjon **programot**, amely a **standard bemenetről** soronként **egész számokat** olvas be! Minden sorban az első szám azt határozza meg, hogy hány további szám található még az adott sorban. A program minden sor esetén írja a **standard kimenetre** a „YES” szót, ha a második számtól kezdve a sor végéig található számok **mindegyike páros**, különben pedig a „NO” szót írja a kimenetre! A bemenetet egy olyan sor zárja, amely csak egyetlen 0-t tartalmaz, ezt a sort a programnak már nem kell feldolgoznia.
5. Írjon **eljárást**, amely **paraméterként** megkap egy **állománynevet**, egy **sztringeket** tartalmazó **egydimenziós tömb** kezdőcímét, valamint a tömb elemszámát! Az eljárás **írja ki** a megadott nevű szöveges **állományba** soronként, a tömbbeli előfordulásuk sorrendjében a tömb azon **elemeit**, amelyek **mondatzáró írásjellel** (pont, kérdőjel, felkiáltójel) **végződnek**! Az eljárás az egyes sztringeket úgy írja ki az állományba, hogy a bennük szereplő angol kisbetűket alakítsa **nagybetűssé**, de az eredeti sztring ne változzon!

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2017. május 27.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

Adott az alábbi struktúra:

```
struct tantargy
{
  char kod[16], *nev;
  int kredit, oraszam; /* mindketto 0 es 100 koze esik */
};
```

1. Írjon konverziós **függvényt**, amely paraméterként megkap egy `struct tantargy` típusú struktúrát, és visszaadja annak a **sztring megfelelőjét**! A sztring elején a tantárgy neve, majd egy szóközt követően zárójelben a kódja, a kreditszáma és az óraszáma szerepeljen egy-egy vesszővel elválasztva! (Segítség: egy sztringbe történő formázott íráshoz lásd az `sprintf()` függvényt.)
2. Írjon **programot**, amely a **standard bemenet** első sorából beolvas egy *n* egész számot ($1 \leq n \leq 100$)! A bemenet következő *n* sorának mindegyike egy legfeljebb 15 karakteres, kizárólag betűkből és számjegyekből álló **tantárgykódot** és egy 0 és 100 közé eső **kreditszámot** tartalmaz egy szóközzel elválasztva! A program írja a **standard kimenetre** a **kreditszámok összegét** külön sorban, majd soronként listázza ki a **tantárgykódokat lexikografikusan növekvő** sorrendben! (Segítség: a rendezéshez lásd a `qsort()` függvényt.)
3. Írjon **programot**, amelynek az első parancssori argumentuma egy **szöveges állomány** neve! Az állomány legalább 1, legfeljebb 200 sort tartalmaz, amelyek felépítése a következő:

   kód;*név*;*kredit*;*óraszám*

   A *kód* egy legfeljebb 15, a *név* egy legfeljebb 40 karakteres, pontosvesszőt nem tartalmazó sztring, a *kredit* és az *óraszám* 100-nál nem nagyobb nemnegatív egész számok. A program írja **a standard kimenetre** soronként, az állományban való előfordulásuk sorrendjében **a legkisebb óraszámmal** rendelkező tantárgyak kódját és nevét egy szóközzel elválasztva! (Segítség: egy sztring egész számmá történő konvertálásához lásd az `atoi()` vagy az `sscanf()`, egy sztring feldarabolásához pedig az `strtok()` függvényt.)
4. Írjon **programot**, amely a **standard bemenetről** soronként tantárgyak adatait olvassa be állományvégjelig! Egy-egy sor felépítése a következő:

   kód;*név*;*kredit*;*óraszám*[;*óraszám*]…

   A *kód* egy legfeljebb 15, a *név* egy legfeljebb 40 karakteres, pontosvesszőt nem tartalmazó sztring, a *kredit* és az *óraszám* 100-nál nem nagyobb nemnegatív egész számok. A tantárgy teljes óraszámát a sorban található óraszámok összege adja meg. A program írja **a standard kimenetre** a **legnagyobb kreditszámmal rendelkező tantárgy nevét**! Ha több ilyen tantárgy is létezne, akkor a bemeneten leghamarabb előforduló tantárgy nevét kell kiírni. Figyeljen arra, hogy a sorok tetszőleges hosszúságúak lehetnek! (Segítség: egy karakter beolvasásához lásd a `getchar()` függvényt.)
5. Írjon **függvényt**, amely paraméterként megkapja egy `struct tantargy` típusú elemeket tartalmazó **kétdimenziós tömb** kezdőcímét, sorainak és oszlopainak számát, valamint egy dupla pontosságú valós számot (*s*)! A függvény az eredeti tömb megváltoztatása nélkül hozzon létre egy **új egydimenziós tömböt**, amely azoknak a **tantárgyaknak a nevét** tartalmazza csupa **nagybetűssé** alakítva, amelyeknél **az óraszám és a kreditszám hányadosa** $s - 1$ és $s + 1$ **közé esik**! A tantárgynevek az eredeti tömb **sorfolytonos** bejárásának sorrendjében szerepeljenek az új tömbben! A tömb **utolsó elemében NULL** pointer szerepeljen, ezzel jelezve a tömb végét! A függvény visszatérési értéke az új tömb kezdőcíme legyen! Ügyeljen arra, hogy C-ben két egész szám hányadosa egész! (Segítség: egy karakter nagybetűssé történő konvertálásához lásd a `toupper()` függvényt.)

Magas szintű programozási nyelvek 1 beugró (levelező tagozat) **2017. május 29.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon **függvényt**, amely **paraméterként** megkap **két sztringet**! A függvény a paramétereként megkapott sztringek megváltoztatása nélkül hozzon létre és **adjon vissza** egy **új sztringet**, amelyet úgy kapunk, hogy **felváltva összefűzzük** az első, illetve a második paraméter sztringben szereplő, egy-egy pontosvesszővel elválasztott **részsztringeket**! Pontosabban: az első paraméter első részsztringjéhez hozzáfűzzük a második paraméter első részsztringjét, majd az első paraméter második részsztringjét, aztán a második paraméter második részsztringjét, és így tovább. Amikor elérjük az egyik sztring végét, akkor a másik sztring fennmaradó részsztringjeit közvetlenül egymás után csatoljuk az eredmény sztringhez. Ha valamelyik megkapott sztringben egyetlen pontosvessző karakter sincs, akkor az egész egyetlen részsztringnek számít. Ügyeljen arra, hogy a részsztringek (és a teljes sztringek) üresek is lehetnek!
2. Írjon **programot**, amely a standard kimenetre írja soronként egyesével a **legrövidebb parancssori argumentumait lexikografikusan csökkenő** sorrendben!
3. Írjon **programot**, amely az első **parancssori argumentumaként** megadott szöveges **állományból** soronként **vizsgák** adatait olvassa be, legalább egyét! Egy-egy sor a következő alakú:

   [*hallgató*[,*hallgató*]…;]*tantárgynév*;*év*.*hónap*.*nap*

   A *tantárgynév* azt mondja meg, hogy melyik tárgyból van meghirdetve a vizsga. A tantárgy neve előtt a vizsgára jelentkezett hallgatók névsora szerepel. Elképzelhető, hogy a vizsgára egyetlen hallgató sem jelentkezett; ebben az esetben a sor mindjárt a tantárgynévvel kezdődik. A sort a vizsga dátuma zárja, amelyben az *év* pontosan négy, a *hónap* és a *nap* pontosan két karakterrel van megadva. A hallgatók nevei egyenként legfeljebb 50, a tantárgy neve legfeljebb 100 karakter hosszúságú. Sem a hallgatók nevei, sem a tantárgynevek nem tartalmaznak sem vessző, sem pontosvessző karaktert. A teljes sor hossza nem haladja meg az 1000 karaktert.

   A program határozza meg és írja a **standard kimenetre a legkésőbbi vizsgához tartozó tantárgy nevét**! Ha a legkésőbbi vizsga napján több vizsga is lenne, akkor közülük a felsorolás szerinti legelsőhöz tartozó tantárgynevet kell a programnak a kimenetre írnia.
4. Írjon **programot**, amely a **standard bemenetről** soronként **egész számokat** olvas be! Minden sorban az első szám azt határozza meg, hogy hány további szám található még az adott sorban. A program minden sor esetén írja a **standard kimenetre** a „YES” szót, ha a második számtól kezdve a sor végéig található számok **mindegyike pozitív**, különben pedig a „NO” szót írja a kimenetre! A bemenetet egy olyan sor zárja, amely csak egyetlen 0-t tartalmaz, ezt a sort a programnak már nem kell feldolgoznia.
5. Írjon **eljárást**, amely **paraméterként** megkap egy **állománynevet**, egy **sztringeket** tartalmazó **egydimenziós tömb** kezdőcímét, valamint a tömb elemszámát! Az eljárás **írja ki** a megadott nevű szöveges **állományba** soronként, a tömbbeli előfordulásuk sorrendjében a tömb azon **elemeit**, amelyekben **van legalább két szóköz egymás mellett**! Az eljárás az egyes sztringeket úgy írja ki az állományba, hogy a bennük szereplő legalább kételemű **szóközcsoportokat** egy-egy **csillag karakterrel helyettesíti**, de az eredeti sztring ne változzon!

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2018. május 18.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely a standard bemenetről egész számokat olvas mindaddig, amíg a 0 számot nem olvassa! A programja minden beolvasott szám (n) esetén nyomtasson a standard kimenetre n sort! Az n sor mindegyikében $2n + 1$ darab karaktert kell megjeleníteni, amelyeknek a csillag karakterei (`'*'`) egy olyan piramist formáznak, melynek az alapja $2n - 1$ hosszúságú, szintjeinek szélessége pedig fokozatosan csökken, egészen a piramis csúcsáig, a példa kimenetben látható módon. A piramis melletti két póznát `'|'` (függőleges vonal) karakterekből rajzolja a kimenetre! A piramisokat egy-egy üres sorral válassza el egymástól úgy, hogy az utolsó piramis után már ne írjon újabb üres sort a standard kimenetre!
2. Írjon függvényt, amely paraméterként megkapja egy sztring kezdőcímét, és visszaadja a sztringben található leghosszabb szóközcsoport (közvetlenül egymás mellett álló szóközök sorozata) hosszát!
3. Egy társasjátékot ketten játszanak. Mindkét játékos két dobókocka segítségével határozza meg azt, hogy mennyit lépnek, amikor rájuk kerül a sor. A játék során felváltva dobnak, összeadják a dobott értékeket, és ennyi lépést tesznek meg egy-egy alkalommal. Abban az esetben azonban, ha valamelyik játékos a két kockával azonos értéket dob, meglépi a dobott számok összegét, majd aztán újra ő dobhat.

   Írjon programot, amely a standard bemenetről állományvégjelig soronként két, egymástól egy szóköz karakterrel elválasztott egész számot olvas be! A két szám az éppen soron következő játékos által dobott két értéket adja meg. A programja írja a standard kimenetre, hogy a két játékos hányat lépett összesen a játék folyamán! Kövesse a példa kimenetben látható formátumot! A játékot mindig az A játékos (`Player A`) kezdi.
4. Írjon programot, amely a standard bemenetről állományvégjelig (EOF-ig) soronként két, egymástól egy szóközzel elválasztott egész számot olvas be, legfeljebb 20 párt! Egy számpár azt írja le, hogy egy jármű hány perc alatt mekkora távolságot tesz meg. Az első szám az eltelt időt mutatja (percekben kifejezve), míg a második a megtett távolságot méterekben számolva. A két adat alapján kiszámítható mindegyik jármű átlagsebessége. A programja írja az átlagsebességeket csökkenő sorrendben a standard kimenetre! A sebességértékeket pontosan két tizedesjeggyel adja meg a példa kimenetben látható formában!
5. Írjon programot, amely a standard bemenetről legfeljebb 20 karakter hosszúságú sztringeket olvas be állományvégjelig (EOF-ig)! A sztringek egy utca képét írják le, ahol az '=' karakter jelzi az utcának azt a részét, ahol nincs kirakva a ház elé szemeteskuka, `'O'` (nagy O betű) pedig azt, ahol igen. A szemétszállítók persze sokkal jobban örülnének, ha nem kellene végigjárniuk az utcát a szemeteskukák ürítése céljából, hanem azokat már eleve odakészítenék nekik az utca valamelyik végébe, szépen egymás mellé rakva őket.

   A programja határozza meg és írja a standard kimenetre, hogy adott bemenet esetén az utcának melyik végébe célszerű a kukákat úgy csoportosítani, hogy ahhoz a lehető legkevesebb kukát kelljen az eredeti helyéről elmozdítani! Ha az utca jobb oldalára, akkor egy `'R'` betűvel, ha balra, akkor egy `'L'` betűvel kezdje a sort, majd egy szóközt követően a mozgatandó kukák számát írja ki! Ha mindkét irányban azonos számú szemeteskuka mozgatásával megoldható a feladat, akkor balra mozgassa a kukákat!
6. Írjon függvényt, amely paraméterként megkap egy sztringet! A függvénynek a paraméterként megadott sztringről el kell döntenie, hogy az jó jelszó-e. Jónak nevezünk egy jelszót, ha tartalmaz kisbetűt, nagybetűt, számjegyet és egyéb karaktert is. Amennyiben a paraméterként megkapott sztring jó jelszó, a függvény hozzon létre egy négyelemű tömböt, benne a kisbetűk, nagybetűk, számjegyek és egyéb karakterek darabszámával (ebben a sorrendben), és adja vissza ennek a tömbnek a kezdőcímét! Ha nem így lenne, akkor a függvény NULL értéket adjon vissza!

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2018. május 22.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely háromszögeket rajzol a standard kimenetre a példában látható módon! A bemenet első sora egy *t* pozitív egész számot tartalmaz, ez a tesztesetek száma. A következő *t* sor mindegyike egy *n* pozitív egész számot tartalmaz. A kimenetre minden tesztesetnél egy *n* hosszúságú oldalakból álló háromszöget kell rajzolni `' '` (szóköz) és `'*'` (csillag) karakterekből. A háromszög derékszögű, egymásra merőleges oldalai az első oszlopban és az első sorban vannak, oldalvonalainak a vastagsága egy egység. Mindegyik sorban közvetlenül soremelés követi a legutoljára kiírt csillag karaktert (azaz nem szerepelnek a sorok végén szóközök). A tesztesetekhez tartozó kimeneteket egy-egy üres sor választja el egymástól. Figyelem: az utolsó háromszöget nem követi üres sor!
2. Írjon függvényt, amely paraméterként megkapja egy sztringeket címző mutatókat tartalmazó tömb kezdőcímét és elemszámát, meghatározza a tömbben szereplő lexikografikusan legkisebb sztringet, és visszaadja, hogy a tömbben hányszor fordul elő ez a sztring!
3. Írjon programot, amely a standard bemenet soraiból egy franciakártya-pakli lapjait olvassa be, soronként egyet-egyet! A kártyalapok színeit a kártyák színeinek angol elnevezése alapján `C` (clubs, treff), `D` (diamonds, káró), `H` (hearts, kőr) és `S` (spades, pikk) betűkkel, értékeit a `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `T`, `J`, `Q`, `K`, `A` számokkal és betűkkel jelöljük. A színt és az értéket egy szóköz karakter választja el egymástól.

   A bemenet első sorából olvasott kártyalap lesz a kezdő lap, ezt a lapot helyezzük az asztalra, és minden ezután lerakott lapot erre a lapra teszünk rá, egy halmot építve így. A további kártyákat a beolvasás sorrendjében próbáljuk meg a halom legfelső lapjához illeszteni. Egy lap akkor illeszthető a halom tetején lévőhöz, ha vagy a színe, vagy az értéke megegyezik azéval (de mindkettő egyszerre nem egyezhet meg vele). Amennyiben az illesztés sikeres, a vizsgált lapot lerakjuk a halom tetejére, innentől kezdve az lesz a halom legfelső lapja. Ha nem lehet a soron következő lapot a halom tetején lévőhöz illeszteni, akkor ez a lap a nyakunkon marad, ezzel a későbbiekben már nem tudunk mit kezdeni. Tevékenységünket mindaddig folytatjuk, míg az összes beolvasott kártyalapot végig nem néztük.

   A programja írja a standard kimenetre azoknak a kártyalapoknak a számát, amelyek a nyakunkon maradtak, azaz amelyeket a pakli végignézése során nem lehetett hozzáilleszteni a fenti szabályok szerint a halom tetején lévő laphoz! Ne felejtse el a számot tartalmazó sort soremelés karakterrel zárni!
4. Írjon programot, amely megállapítja, hogy a parancssori argumentumai hosszuk szerint monoton sorozatot alkotnak-e, feltételezve, hogy a programnak legalább egy parancssori argumentuma van! Ha igen, akkor a programnak egy „`YES`”, egyébként egy „`NO`” üzenetet kell a standard kimenetre írnia.
5. A következő kirándulásunkhoz a Google Maps készített útvonalajánlatot. Az ajánlaton egyenlőségjelek (`'='`) jelzik az utat, nagybetű karakterek a köztes csomópontokat, kis `'o'` karakterek pedig az út során érintett benzinkutakat.

   Írjon programot, amely a standard bemenet első sorából egy legfeljebb 50 karakter hosszúságú sztringet olvas be állományvégjelig (EOF-ig)! A sztring a Google Maps egy, a fentebb leírtak alapján értelmezhető útvonalterve. A bemenet további sorai csomópont-azonosító nagybetűket tartalmaznak, soronként mindig két különbözőt, egy-egy szóközzel elválasztva. Ezek a nagybetűk biztosan előfordulnak az útvonaltervben is. A programjának azt kell meghatároznia, hogy a megadott csomópontokat végpontokként tekintve melyik irányból érhető el a legközelebbi benzinkút, ha az egyik megadott csomópontból a másik felé indulunk el! A kimenetre a kiindulásnak javasolt csomópontot és az elsőként érintett benzinkút távolságát kell kiírnia. Ha mindkét irányból azonos távolságra esik a legközelebbi benzinkút, akkor az elsőként megadott csomópont azonosítóját kell a kimenetre írnia! Ha nem lenne a két csomópont között egyetlen benzinkút sem, akkor a „`No gas station!`” szöveget írja a standard kimenetre!
6. Írjon függvényt, amely paraméterként megkapja egy egész számokat tartalmazó tömb kezdőcímét és a tömb elemeinek számát! A függvény – az eredeti tömb módosítása nélkül – hozzon létre egy új, egész számokat tartalmazó tömböt, amely az eredeti tömb elemeinek a legnagyobb elemtől való eltérését tartalmazza!

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2018. május 26.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely egy üres, 8×8-as sakktáblát rajzol a standard kimenetre! A világos mezőket '+' karakterekkel, a sötét mezőket '*' karakterekkel töltse fel! A program bemenete egy sorból áll, amelyben két, egymástól egy szóközzel elválasztott egész szám található. Ezek a számok 1 és 100 közé esnek, és megmutatják, hogy egy-egy mező a karakteres kimeneten rendre hány oszlopból és sorból áll. A sakktábla bal felső sarka mindig világos mező.
2. Írjon függvényt, amely paraméterként megkapja egy sztringeket címző mutatókat tartalmazó tömb kezdőcímét és elemszámát, és visszaadja a tömbben szereplő, érvényes MAC-címet tartalmazó sztringek számát! Egy MAC-címet akkor tekintünk érvényesnek, ha pontosan 12 darab hexadecimális számjegyből áll (a betű karakterek lehetnek kis- és nagybetűk is), amelyeket kettesével vagy csupa '-' karakterek, vagy csupa ':' karakterek tagolnak.
3. Írjon programot, amely a bemenetére érkező műveleteket soronként elvégzi, és eredményüket szintén soronként a kimenetre írja! Minden művelet binér művelet: összeadás ('+'), kivonás ('-'), szorzás ('*') vagy egész osztás ('/'). A műveleti jelet az operandusoktól egy darab szóköz választja el. Egy-egy operandus vagy egy nemnegatív decimális egész, vagy az előző művelet eredménye lehet. Utóbbit 'x' jelöli. Ha az 'x' már az első sorban előfordulna, akkor az 0 értéket takar. Ha 0-val való osztás történne, akkor az adott művelet esetén a „Hiba!” szöveget kell a kimenetre írni, és az előző művelet eredményét visszük tovább a következő műveletre is.
4. Írjon programot, amely megadja a bemenetére érkező legfeljebb 1000 darab, szóközökkel és újsor karakterekkel tagolt, –1000 és +1000 közé eső egészek közül a 10 legkisebbet! Ha egy szám a bemeneten többször is megjelenik, akkor azt csak egyszer vegye figyelembe! A kimeneten egyetlen sorban, szóközzel elválasztva, növekvő sorrendben adja meg a számokat! Ha nincs 10 különböző szám a bemeneten, akkor a program írja ki mindegyiket!
5. Írjon programot, amely meghatározza az alábbi kifejezés értékét:

$$\sum_{j=1}^{5} \left| a_j^3 - \sum_{i=1}^{4} (a_i - a_{i+1})^2 \right|$$

   A bemenet minden sora rendre az a_1, a_2, a_3, a_4, a_5 változók értékeit tartalmazza egy-egy szóközzel elválasztva. Ezen változók értéke –100 és +100 közé eső egész szám. A kimenetre minden bemeneti sorra egyetlen sort kell kiírni, amelyben a fenti kifejezés értékét kell megadni.
6. Egy üzenetet úgy is titkosíthatunk, hogy betűit egy hosszabb szövegbe rejtjük. A titkos kulcs ekkor azt mutatja meg, hogy a szöveg mely karakterei tartoznak az üzenethez.

   Írjon függvényt, amely paraméterként megkap két sztringet: egy kódolt szöveget és egy kulcsot, előállít egy új sztringet, amely a dekódolt szöveget tartalmazza, végül visszaadja az új sztring kezdőcímét! A két paraméter azonos hosszúságú sztringek, és legfeljebb 1000 karakter hosszúságúak. A kulcs '+' és '.' karakterekből áll. A kulcs '+' karakterei azon karakterpozíciókat jelölik, ahol a kódolt szövegben értékes karakter található. A visszaadott sztring a kódolt szöveg azon karaktereiből áll össze, amelyek pozíciói megegyeznek a kulcsban lévő '+' karakterek pozícióival (balról jobbra haladva).

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2019. május 20.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely megadott amplitúdójú és frekvenciájú, háromszög alakú hullámformákat állít elő! A bemenet egy pozitív egész számmal kezdődik, amely a tesztesetek számát jelenti. Ezt követi egy üres sor, majd a tesztesetek, szintén egy-egy üres sorral elválasztva. Minden teszteset két egész számból áll, amelyek külön sorban vannak megadva. Az első az amplitúdó, a második a frekvencia. A kimenetnek minden teszteset esetén az alább leírtak szerint kell kinéznie. Két egymást követő teszteset kimenetét egy-egy üres sorral kell egymástól elválasztani. Minden tesztesetre hullámformákat kell kinyomtatni, szintén egy-egy üres sorral elválasztva egymástól. A hullámformák darabszáma egyenlő a frekvenciával, míg minden hullám „magassága” egyenlő az amplitúdóval. Az amplitúdó sohasem lesz nagyobb kilencnél. Magának a hullámformának minden egyes sorát azzal az egész számmal kell feltölteni, amely az adott sornak a „magasságát” jelzi.
2. Írjon függvényt, amely a paramétereként megkapott tetszőleges hosszúságú sztringről eldönti, hogy az egy szabályos útvonalleírást tartalmaz-e! Szabályosnak tekintünk egy útvonalleírást, ha az a következő alakú:

   településnév – *településnév*[– *településnév*]...

   A településnév minden esetben egy olyan karaktersorozat, amely kizárólag betű és kötőjel (mínuszjel) karaktereket tartalmaz. Két településnév között pontosan három karakter található: egy szóköz, egy kötőjel (mínuszjel) és egy újabb szóköz. A szabályos útvonalleírás legalább két településnevet tartalmaz. Ha a sztring ilyen alakú, akkor a függvény igaz értéket, egyébként hamisat adjon vissza!
3. Írjon programot, amely a standard bemenetről állományvégjelig (EOF) soronként egész számokat olvas be! A sorok első száma (n), amelynek értéke minden esetben 1 és 20 közé esik (beleértve ezt a két számot is), a sor további elemeinek a darabszámát adja meg. A programja minden sor esetén határozza meg, hogy a sor első elemét nem számolva, a többi elem vajon az 1, 2, 3, …, n számok egy permutációját alkotják-e! Ha az adott értékek az 1, 2, 3, …, n számok egy permutációját alkotják, akkor a programjának a „`YES`” szót, ellenkező esetben a „`NO`” sztringet kell a kimenetre írnia.
4. Írjon programot, amely parancssori argumentumait hosszuk szerint növekvő sorrendben írja a standard kimenetre soronként, feltételezve, hogy minden parancssori argumentum hossza különböző!
5. Írjon programot, amely a standard bemenetről soronként egy-egy legfeljebb 30 karakter hosszúságú sztringet olvas be mindaddig, amíg a „`THE END`” sztringet nem olvassa! Minden sztring esetében határozza meg, hogy hány angol ábécébeli nagy- és kisbetűt tartalmaz, majd írja a standard kimenetre a nagy- és kisbetűk darabszámát, valamint az „`UPPERCASE`”, a „`LOWERCASE`” vagy az „`EQUALS`” szavak egyikét attól függően, hogy a sztringben nagybetűből vagy kisbetűből van-e több, esetleg azonos-e ezeknek a darabszáma!
6. Írjon függvényt, amely paraméterként megkap egy páros darabszámú karakterből álló nemüres sztringet, amely betű és számjegy karaktereket tartalmaz váltakozó sorrendben! Ebben a sztringben minden számjegy karakter azt jelzi, hogy az eredeti üzenetben hányszor ismétlődött a számjegy karaktert közvetlenül megelőző betű karakter. A függvény állítsa elő az eredeti üzenetet tartalmazó sztringet, és adja vissza annak kezdőcímét!

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2019. május 22.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely a standard bemenetről soronként négy pozitív egész számot olvas be! A program az egy sorban szereplő négy szám közül az első két számot egy tört számlálójaként és nevezőjeként értelmezze (ebben a sorrendben), a harmadik és negyedik számot pedig egy másik tört számlálójaként és nevezőjeként értelmezze (ebben a sorrendben)! A program írjon a standard kimenetre soronként egy '<' (kisebb), '>' (nagyobb) vagy '=' (egyenlőségjel) karaktert attól függően, hogy az első tört értéke kisebb vagy nagyobb a második tört értékénél, vagy esetleg egyenlő vele! A bemenet végét egy olyan sor jelzi, amely négy darab 0-s számot tartalmaz, ezt a sort a programnak már nem kell feldolgoznia.

2. Mint ismeretes, a Télapó minden évben december 6-án este meglátogatja a gyerekeket, jókat és rosszakat egyaránt, és megjutalmazza őket attól függően, hogy mennyi C nyelvű programot írtak a legutóbbi látogatása óta eltelt időben.

   Írjon függvényt, amely paraméterként megkapja egy érvényes dátum három egész értékű komponensét: egy évszámot, egy hónapszámot és egy napszámot, ebben a sorrendben! A függvény visszatérési értéke legyen az az egész szám, amely jelzi, hogy mennyit kell még aludni a Télapó következő látogatásáig (1 alvás/éjszaka értékkel számolva és a napközbeni alvásoktól eltekintve)! Számításai során vegye figyelembe a csak szökőévekben előforduló február 29-i napokat is! Egy év akkor szökőév, ha osztható 4-gyel, de nem osztható 100-zal, vagy osztható 400-zal.

3. Írjon programot, amely soronként egész számokat olvas be, és eldönti róluk, hogy két megadott érték közé esnek-e az értékeik! A bemenet első sora pontosan egy darab pozitív egész számot (N) tartalmaz, ami a feldolgozandó tesztesetek számát jelzi. A következő N sor mindegyikében egész számok szerepelnek. Minden sor első száma (K, ahol $3 \leq K \leq 10$) azt árulja el, hogy hány további szám található a sorban. Programjának azt kell eldöntenie, hogy a további számok mindegyike a sorban másodikként és utolsóként beolvasott két érték közé esik-e, megengedve a két értékkel való egyezést is! Ha igen, akkor a programnak egy „`YES`”, egyébként egy „`NO`” üzenetet kell a standard kimenetre írnia minden feldolgozott tesztesetre.

4. Írjon programot, amely soronként öt darab egész számot olvas be állományvégjelig (EOF)! Az öt szám franciakártya-lapok értékeit jelzi: az 1-es az ász, a 11-es a bubi, a 12-es dáma, a 13-as a király, a többi érték saját magát képviseli. A program minden sor esetén döntse el, hogy van-e négy egyforma érték (póker) a beolvasott öt szám között, és írja ki soronként a „`YES`” szöveget a standard kimenetre, ha van, illetve a „`NO`” szöveget, ha nincs!

5. Írjon programot, amely a standard bemenetről soronként egy-egy személy nevét, lakásának települését és annak irányítószámát, valamint fizetését olvassa be! A felsorolt adatok egymástól pontosvessző karakterrel vannak elválasztva. Egyik sor hossza sem haladja meg a 120 karaktert. A program határozza meg és írja a standard kimenetre soronként azoknak a személyeknek a nevét, akik a 4032-es irányítószámú településen laknak!

6. Írjon függvényt, amely paraméterként megkap egy nemüres sztringet, amely kizárólag az angol ábécé betűiből áll úgy, hogy azonos betűkből legfeljebb 9 darab szerepel közvetlenül egymás mellett a sztringben! A függvény állítsa elő ennek a sztringnek a futamhosszkódolt változatát, és adja vissza annak kezdőcímét! Egy sztringből annak futamhosszkódolt változatát úgy állíthatjuk elő, hogy az azonos betűkből álló betűcsoportokat a csoport egy betűjére és a csoportban lévő betűk darabszámát megadó számjegy karakterre cseréljük le.

Magas szintű programozási nyelvek 1 / Programozási nyelvek 1 / Programozás 1

gyakorlati számonkérés (levelező tagozat) **2019. május 29.**

A megoldásaiban ügyeljen arra, hogy minden programjának 0 legyen a visszatérési értéke, és minden kimenetre kiírt sort újsor karakterrel zárjon! A feladatok pontos leírását példákkal a ProgCont honlapján találja.

1. Írjon programot, amely a standard bemenetről egész számokat olvas be állományvégjelig, majd meghatározza és a standard kimenetre írja a leghosszabb olyan számsorozatnak a hosszát, amely monoton növekvő sorozatot alkot! A bemeneten az egész számokat fehér karakterek (szóközök, tabulátorok és újsor karakterek) választják el egymástól.
2. Írjon logikai függvényt, amely paraméterként megkap egy sztringet, és igazat ad vissza, ha a benne előforduló különböző karakterek mindegyike legalább kétszer szerepel a sztringben, ellenkező esetben pedig hamissal tér vissza!
3. Írjon programot, amely tömbök rendezetlenségi értékét határozza meg és írja a standard kimenetre! Egy tömb rendezetlenségi értékének meghatározását a tömbelemek összehasonlításával végezzük: növekvő rendezettséget feltételezve az A tömb i-edik és j-edik eleme egymáshoz képest rendezetlen, ha $i < j$ esetén $A[i] > A[j]$. A teljes tömbre vonatkozó értéket az összes tömbelem összes többivel történő összehasonlításával és a rendezetlen esetek összeszámolásával kapjuk meg.

   A bemenet első sora pontosan egy darab pozitív egész számot (N) tartalmaz, ami a feldolgozandó tesztesetek számát jelzi. A következő N sor mindegyikében pozitív egész számok szerepelnek. Minden sor első száma (K, ahol $1 \leq K \leq 10$) azt árulja el, hogy hány eleme lesz a tömbnek. A további számok a sorban a tömb elemei. Minden tesztesetre soronként egyetlen számot kell kiírni, a feldolgozott tömb rendezetlenségi értékét.
4. A prímgyűrű nagyon értékes eljegyzési ajándék. Nagy ritkaságnak számít, és emiatt fokozottan oda szoktak rá figyelni. Egy olyan gyűrűről van szó, amelynek követ pozitív egész számok alkotják úgy, hogy bármely két, közvetlenül egymás szomszédságában lévő szám összegének prímszámnak kell lennie. (A prímszám olyan szám, amelynek – eltekintve az előjeltől – az 1-en és önmagán kívül nincs más egész osztója.) Ráadásul, ha egy prímgyűrű n darab követ tartalmaz, akkor a kövek között kell szerepelnie 1-től n-ig az összes pozitív egész számnak. Ha belegondolunk, ilyen feltételek mellett például nem is lehet páratlan darabszámú kőből álló prímgyűrűt készíteni.

   Írjon programot, amelynek legalább kettő darab parancssori argumentuma van, mindegyik egy pozitív egész szám! A program döntse el, hogy alkothatnak-e a megadott sorrendben a parancssori argumentumok prímgyűrűt vagy sem, és írja a standard kimenetre a prímgyűrű értékét (az őt alkotó számok összegét), valamint tőle pontosan egy szóköz karakterrel elválasztva a „`YES`” literált, ha igen, vagy csupán a „`NO`” literált, ha nem! Ne felejtsen el a kiírt sztring mögé közvetlenül egy soremelés karaktert írni!
5. Mitől erős egy jelszó? Nehéz a kérdésre egyértelmű választ adni, de általában elvárás egy erős jelszóval szemben, hogy viszonylag hosszú legyen, és sok különböző típusú karaktert tartalmazzon: kisbetűt, nagybetűt, decimális számjegyet és egyéb, az előzőek közé nem sorolható karaktert.

   Írjon programot, amely a standard bemenetről állományvégjelig soronként egy-egy sztringet olvas be, és eldönti a beolvasott sztringről, hogy az megfelel-e a fenti kritériumnak! Ha úgy találja, hogy igen, azaz a sztring legalább 6 karakter hosszúságú, továbbá tartalmaz kisbetűt, nagybetűt, decimális számjegyet és egyéb, az előzőek közé nem sorolható karaktert is, akkor a standard kimenetre írja a „`GOOD`” szót, ha pedig nem, akkor a „`BAD`”-et!
6. A leginkább átfogó vallási-kulturális reformjáról, az ún. Amarna-reformról ismert Ehnaton fáraó (eredeti nevén IV. Amenhotep) emlékét utódai, amennyire csak tudták, mindenhonnan eltüntették: nevét kivájták az épületek köveiből, leradírozták a kartusokról. Az ön feladata ennek a folyamatnak a modellezése lesz ebben a feladatban.

   Írjon függvényt, amely paraméterként megkap két sztringet, és előállít egy olyan új sztringet, amelyet úgy kapunk, hogy az első paraméterként megkapott sztringben megkeressük a másodikként megkapott sztring összes előfordulását, és helyettesítjük azokat egy-egy „`***`” sztringgel! A függvény adja vissza ennek az új sztringnek a kezdőcímét! Ügyeljen rá, hogy az eredeti sztring ne módosuljon!