Informatika 2 második zárthelyi, 2009. május 4.

Két legkisebb

A következő (kissé hibás) program beolvas a bemenetről pontosan 14 darab egész számot (feltesszük, hogy mindegyik 0 és 999 közé esik), és kiírja közülük a legkisebb és a második legkisebb számot.

```
 1 #include <stdio.h>
 2 int
 3 main(void) {
 4         int a[14], k, mp, mv, m2v;
 5         for (k = 0; k < 14; k++)
 6                 scanf("%d", &a[k]);
 7         mp = -1; mv = 10000;
 8         for (k = 0; k < 14; k++)
 9                 if (a[k] < mv) {
10                         mp = k;
11                         mv = a[k];
12                 }
13         m2v = 10000;
14         for (k = 0; k < 14; k++)
15                 if (mp != k && a[k] < m2v)
16                         m2v = a[k];
17         printf("%d %d\n", mv, m2v)
18         return 0;
19 }
```

Például ha a bemenet a következő:

```
92 21 87 57 99 63 22 25 3 61 34 87 86 18
```

akkor a kimenet:

```
3 18
```

1. feladat

Ha megpróbáljuk lefordítani a fenti programot a `gcc -Wall -O -o z2g1 z2g1.c`, a fordító az alábbi hibaüzenetet adja, és nem állít elő futtatható programot.

```
z2g1.c: In function 'main':
z2g1.c:18: error: expected ';' before 'return'
```

Javítsa ki az apró hibát a kódban.

2. feladat

A 7–12. sorok kiszámolják a legkisebb számot (`mv`) és ennek indexét a tömbben (`mp`). Miért kell mindkettőt kiszámolnunk? Meg lehetne-e oldani, hogy csak az egyiket tároljuk el? (A programnak úgy kell működnie, hogy ha a legkisebb szám kétszer szerepel a bemenetben, akkor kétszer írjuk ki.)

Kefir

Elővettem a hűtőszekrényből egy doboz kefirt, és szeretném eldönteni, friss-e még, vagy már lejárt a szavatossága. Ehhez leolvassuk a dobozról a szavatossági dátum hónapját és napját, és ismerjük a mai dátumot is. A kefir lejárt, ha a szavatossági dátum már elmúlt, ha viszont a szavatossági dátum egyenlő a mai dátummal vagy későbbi, akkor a kefir még friss.

A következő C nyelvű `friss` függvény megpróbálja eldönteni, hogy a kefir még friss-e. A függvénynek négy egész paramétere van: az első kettő a mai dátum hónap és nap sorszáma, a következő kettő a kefir zárófóliáján látható hónap és nap. A függvény 1-et ad vissza, ha még megehetjük a kefirt, 0-t, ha már megromlott.

```
20 int friss(int most_honap, int most_nap, int folia_honap, int folia_nap) {
21         if (most_honap < folia_honap)
22                 return 1;
23         else if (folia_honap < most_honap)
24                 return 0;
25         else
26                 return most_nap <= folia_nap;
27 }
```

Így például a `friss(4, 11, 4, 18)` hívás visszatérési értéke 1, mert ha ma április 4. van, és a kefir április 18-án jár le, akkor még friss; viszont a `friss(9, 12, 8, 28)` eredménye 0, mert a kefir már két hete lejárt. (Vegye észre, hogy a 7. sor akkor fut le, ha a lejárati dátum az aktuális naptári hónapba esik, és ebben a `most_nap <= folia_nap` összehasonlítás az 1 értéket adja, ha a lejárati dátum napja nagyobb vagy egyenlő a mai dátum napjánál, viszont a 0 értéket adja ha a lejárati dátum napja kisebb.)

A fent megadott implementációnak azonban van egy hibája: feltételezi, hogy a lejárati dátum és a mai dátum ugyanabban a naptári évben van. Például a `friss(2, 4, 12, 27)` a fenti implementáció hibásan az 1 eredményt adja, mert a folyó év december hónapja később van, mint február. A helyes eredmény 0 lenne, mivel a kefir nyilván múlt év decemberében járt le. Hasonlóan a `friss(12, 15, 1, 2)` hívás helyes eredménye 1, mert a kefir jövő év elejéig jó, de a fenti kód hibásan 0-t ad.

A kefires dobozra nincs ráírva a lejárat éve, ezért ezt nem tudjuk összehasonlítani – szerencsére erre nincs is szükség, mivel felteheti, hogy a kefir biztosan nem járt le két hónapnál hosszabb ideje, és viszont nem állhat el mától számított egy hónapnál hosszabban sem. Ezek alapján csak a hónapból és napból meg lehet állapítani, hogy jó-e még a kefir.

3. feladat

Írjon olyan függvényt, ami a fent megadottnak megfelelően viselkedik, de újév táján is pontos eredményt ad.

P betűs szavak

A következő kód beolvas egyetlen hosszú sort, amelyben sok (angol) szó van felsorolva, és kiírja azokat a p betűvel kezdődő szavakat, amelyek egy legalább három egymás utáni, csupa p betűvel kezdődő szavakból álló sorozatban vannak.

```
1  words = readline().split()
2  count = 0
3  (0 ... words.size()).each() {|k|
4          if ?p == words[k][0]
5                  count += 1
6                  if 3 == count
7                          print(words[k - 2], " ", words[k - 1], " ")
8                  end
9                  if 3 <= count
10                         print(words[k], " ")
11                 end
12         else
13                 count = 0
14         end
15 }
16 print "\n"
```

Például ha a bemenet a következő (egy hosszú sor):

```
goldenly pastry posse primers cancerous poppies humidly polo possession instantaneously
doubtless polymer parameterizes policy plays proclaimers precocious penalize trembles
assistances persevered palms pike
```

akkor a kimenet a következő:

```
pastry posse primers polymer parameterizes policy plays proclaimers precocious penalize
persevered palms pike
```

A poppies szó nem szerepel a kimenetben, mert a szomszédai nem kezdődnek p betűvel, de még a polo és posession szavak sem, mert ez csak két egymás utáni p betűs szó, mi pedig legalább három szóból álló sorozatot keresünk.

Némi magyarázat. A kód 1. sora a `words` változót beállítja egy tömbre, aminek egyes elemei a bemenet szavai, így az első elemének `words[0]`-nek az értéke a `"goldenly"` sztring. A 4. sorban a `words[k][0]` egy ilyen szó első karakterének a kódját számolja ki, `?p` pedig a p betű karakterkódja. Az each függvény a 3. sorban blokkjában (a 4–14. sorokban) szereplő utasításokat hajtja végre annyiszor, ahány szót beolvastunk, a `k` változót az első alkalommal 0-ra állítva, majd 1-re, stb. Ha a print függvénynek több sztringet adunk át argumentumként, akkor mindegyiket kiírja egymás után.

4. feladat

Magyarázza meg, miért van szükség két külön feltételes utasításra a 6–11. sorokban? (Gondolja meg, miért működne másképpen a program, ha a 7. és a 10. sor utasítását egyaránt csak egy feltételes utasításba raknánk, amelynek a feltétele a jelenlegi 6. vagy 9. sorrral egyezne meg.)

5. feladat

Hogyan kéne módosítani a kódot, ha nem legalább három, hanem legalább négy p betűs szóból álló sorozatokat keresnénk?

6. feladat

Lehetséges-e, hogy a 7. sorban hibásan negatív indexszel próbáljuk indexelni a words tömböt, pl. ha k értéke 0? Milyen bemenet esetén lehetséges ez, vagy miért nem?