naponta 0,669 mg Pb kerül. Ez a mennyiség már elegendő a krónikus toxicitás kialakulásához!

A mérések eredményeiből az is kitűnik, hogy a legtöbb ólmot a vizsgált növények közül a sárgarépa tartalmazza, legkevesebbet a bab, vagyis a gyökérzöldségek nagyobb mennyiségű ólmot vesznek fel környezetükből, mint a magtermők.

A szennyezés nem a szennyező forrás közelében érvényesül erősebben (a meteorológiai viszonyok következményeként). Az ipari füsttel kibocsátott szennyezőanyag mennyisége időben, 2009-ről 2010-re csökkent .

Nekem megnyugtató következtetés volt, hogy az édesanyám bablevese nem egészségtelen, amennyiben nem főz bele sárgarépát!

Forrásanyag:

Literáthy Péter (szerk.), *Felszíni vizek nehézfém szennyezései*, Műszaki Kk., Bp. 1982.
Simon László: *Nehézfémekkel szennyezett talaj és víz fitoremediációja*, Nyíregyházi Főisk. Táj- és Környezetgazdálkodási tanszék, 2006.
AMAP (1998). *Arctic Monitoring and Assessment Programme assessment report: Arctic pollution issues.* Oslo
de Temmerman L, Hoenig M.: *J. of Atmospheric Chem.*, (2004). 49:121-135.

Szilágyi Renáta
Bethlen Gábor Kollégium, XII. oszt. tanuló (2010/2011. tanév)
témavezető tanár: Dr. László Enikő, Zsigmond Andrea, egyetemi adj.

Érdekes informatika feladatok

XXXX. rész

Egy „fogas kérdés"

A tanévkezdés előtt egy fogast szerettem volna készíteni a négyfalusi Zajzoni Rab István Középiskolában működő napközi óvoda számára. Ekkor találkoztam azzal a problémával, amelyet általánosan így fogalmazhatunk meg:

Adott egy L *hosszúságú fogasléc, amelynek két végén* H *távolságnyira van egy-egy felfogató lyuk, majd ezektől egyenlő távolságnyira még* N *darab felfogató lyuk.*

Adjuk meg, hogy minimum K *darab akasztót hogyan tudunk felszerelni egyenlő távolságra és egyenletesen a fogasra úgy, hogy figyelembe vesszük a felfogató lyukakat: adjuk meg az akasztók számát és azt, hogy a fogasléc bal végétől számítva milyen távolságokra kell felszerelni őket.*

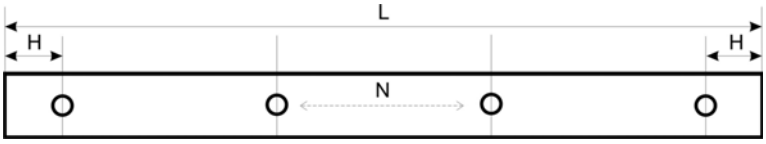
BE: *fogas.in*: négy szám: L H N K (az L és a H valós, az N és a K egész)

KI: *képernyő*: az akasztók száma és új sorokban a távolságok

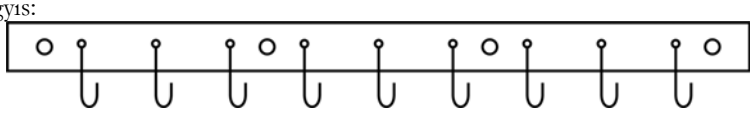
Példa:

200.00 10.00 2 8

Ez a következő lécet feltételezi:



Ezekre a bemeneti adatokra a megoldás:
9
20.00 40.00 60.00 80.00 100.00 120.00 140.00 160.00 180.00
Vagyis:



Láthatjuk, hogy 8 akasztót nem lehet egyenletesen felszerelni a lyukak miatt, csak 9-et, mégpedig úgy, hogy a bal szélétől kezdve a lécnek 20 majd 40, 60, 80 és így tovább távolságokra kell ezeket felszerelni, az utolsót 180-ra.

Hogyan jutottunk ehhez a megoldáshoz? Elemezzük ki a feladatot!

Első lépésként olvassuk be az adatokat:

```
#include<stdio.h>
#include<math.h>

int main()
{
  float L, H;
  int N, K;
  FILE *f = fopen("fogas.in", "r");
  if (f==NULL)
  {
    printf("Allomany hiba (fogas.in)!");
    return 0;
  }
  fscanf(f, "%f", &L);
  fscanf(f, "%f", &H);
  fscanf(f, "%i", &N);
  fscanf(f, "%i", &K);
  fclose(f);
```

Annak a feltétele, hogy az akasztók egyenlő távolságra és egyenletesen legyenek felszerelve a fogasra úgy, hogy figyelembe vesszük a felfogató lyukakat az, hogy a K osztható legyen N+1-gyel, vagyis addig kell növelni a K-t, amíg ez be nem következik:

```
while(K%(N+1) != 0) ++K;
```

Ekkor, mivel az L – 2*H szakaszt kell K egyenlő részre felosztani, legyen:

```
float r = (L-2*H)/K;
```

Így egyszerűen kiszámíthatjuk az első akasztó pozícióját, mégpedig úgy, hogy az első lyuktól (amely H távolságnyira van a fogasléc bal szélétől) `r/2` távolságnyira tesszük (így veszik mindig közre az egymástól `r` távolságnyira lévő akasztók a lyukakat):

```
float kezdo = (H + r/2);
```

Ha felületesen szemlélnénk a feladatot, azt mondhatnánk, hogy már meg is vagyunk, megvan a kezdőpozíció, most már elegendő, ha mindig egymástól `r` távolságnyira kiírnánk az akasztók helyzetét a fogasléc bal széléhez viszonyítva (azt, hogy hány akasztó kell, már kiszámítottuk, ez lett az új K).

Igen ám, de az egyenletes elosztás azt feltételezi, hogy alaposabban meg kell vizsgálnunk a H előtti részt is, hisz ha ez nagyobb, mint `r/2`, akkor ide is kell hogy kerüljön akasztó, mert csak így lesznek egyenletesen elosztva ezek a fogaslécen, sőt lehet, hogy több akasztó is kell ide, és ha a H előtti részre akasztók kellenek, akkor világos, hogy a fogas jobb oldalára, az utolsó lyuk utánra is ugyanennyi akasztó kell, mert a H részek szimmetrikusak.

Vegyük ezekre az esetekre a következő két példát:

204.00 12.00 2 8

és

384.00 102.00 2 8

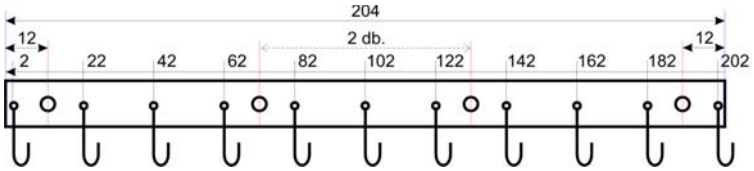
Az első esetében a H részekre csak egy-egy akasztó kell, a második esetben több.

Az első megoldása:

11

2.00 22.00 42.00 62.00 82.00 102.00 122.00 142.00 162.00 182.00 202.00

Tehát:



Míg a másodiké:

19

12.00 32.00 52.00 72.00 92.00 112.00 132.00 152.00 172.00 192.00 212.00 232.00 252.00 272.00 292.00 312.00 332.00 352.00 372.00

A programot tehát így folytatjuk:

```
while(kezdo-r > 0)
{
  K += 2;
  kezdo -= r;
}
```

Vagyis 2-vel növeljük az akasztók számát (a H részek szimmetriája miatt), és eltoljuk `-r`-el a kezdőpozíciót.

Végül nem marad más hátra, mint kiírni az akasztók számát (K), valamint a kezdőpozíciótól számítva az akasztók helyét:

```
printf("%i\n", K);
int j;
for(j=0; j<K; ++j)
{
  printf("%.2f ", kezdo);
  kezdo += r;
}
```

```
  printf("\n");
  return 0;
}
```

A feladat ezzel meg lenne oldva, ha „végtelen” számú akasztó állna a rendelkezésünkre a fogas elkészítése végett, de sajnos a tapasztalat azt mutatja, hogy pont az akasztók a fogas legdrágább részei, így nyilvánvaló, hogy optimalizálnunk kell a számukat illetően, vagyis ha minimum K akasztót kell felszerelnünk, akkor a végső K érték a lehető legközelebb álljon a kezdeti K értékhez. Például az utolsó esetben láttuk, hogy a kezdeti 8 akasztó helyett 19-et kellett felszerelni, ami 11-gyel több, mint az eredeti érték.

Az alapfeladat tehát tovább bővíthető a következőképpen:

Adott egy L *hosszúságú fogasléc, amelynek két végén* H *távolságnyira van egy-egy felfogató lyuk, majd ezektől egyenlő távolságnyira még* N *darab felfogató lyuk.*

Adjuk meg, hogy minimum K *darab akasztót hogyan tudunk felszerelni egyenlő távolságra és egyenletesen a fogasra úgy, hogy figyelembe vesszük a felfogató lyukakat,* ***és úgy, hogy a lehető legkevesebb akasztót használjunk****: adjuk meg az akasztók számát és azt, hogy a fogasléc bal végétől számítva milyen távolságokra kell felszerelni őket.*

BE: *fogas.in*: négy szám: L H N K (az L és a H valós, az N és a K egész)

KI: *képernyő*: az akasztók száma és új sorokban a távolságok

Ha ezt a feladatot szeretnénk megoldani, akkor meg kell vizsgálnunk, hogy ha az `r` részeken csökkentjük az akasztók számát, hogyan viselkedik a program kimenetele. Például, ha az utolsó esetben nem három, hanem csak két akasztóval számolunk az `r` részeken, akkor a kért minimum 8 akasztó helyett nem 19-et, hanem csak 12-őt kell felszerelni, és ez a feladat helyes megoldása.

Habár többféleképpen lehet megoldani ezt az optimalizálási feladatot, mi most úgy oldjuk meg, hogy visszavezetjük az előzőre, vagyis először megoldjuk az első eset szerint, majd csökkentjük az `r` részeken az akasztók számát, újra megoldjuk és minimumot számítunk a kapott K értékek között úgy, hogy ezek még nagyobbak legyenek, mint a kezdeti K. Azt a megoldást fogadjuk el, amely legközelebb van a kezdeti K értékhez:

```
#include<stdio.h>
#include<math.h>

void megold(float L, float H, int N, int &K, int &kk, float
&r, float &kezdo)
{
  while(K % (N+1) != 0) ++K;
  kk=K;
  r = (L-2*H)/K;
  kezdo = (H + r/2);
  while(kezdo-r > 0)
  {
    K += 2;
    kezdo -= r;
  }
}

int main()
{
```

```
    float L, H;
    int N, K;
    FILE *f = fopen("fogas.in", "r");
    if (f==NULL)
    {
      printf("Allomany hiba (fogas.in)!");
      return 0;
    }
    fscanf(f, "%f", &L);
    fscanf(f, "%f", &H);
    fscanf(f, "%i", &N);
    fscanf(f, "%i", &K);
    fclose(f);

    float r, kezdo;
    int kezdetiK = K;
    int kk;
    megold(L, H, N, K, kk, r, kezdo);

    while(kk > kezdetiK)
    {
      K = kk-(N+1);
      megold(L, H, N, K, kk, r, kezdo);
    }

    printf("%i\n", K);
    int j;
    for(j=0; j<K; ++j)
    {
      printf("%.2f ", kezdo);
      kezdo += r;
    }
    printf("\n\n");

    return 0;
  }
```

A megoldás így:
12
27.00 57.00 87.00 117.00 147.00 177.00 207.00 237.00 267.00 297.00 327.00 357.00

Házi feladat

Szerkesszünk egy olyan példát (bemeneti állományt), amely egynél több iterációval csökkenti a K értéket ahhoz, hogy a megoldás optimális legyen!

Kovács Lehel István

Tények, érdekességek az informatika világából

Android verziók

- Az Android egy Linux kernel fölött futó, mobil telefonokra és készülékekre (pl. Tablet PC) írt operációs rendszer. Fejlesztését az *Android, Inc.* kezdte meg, amit