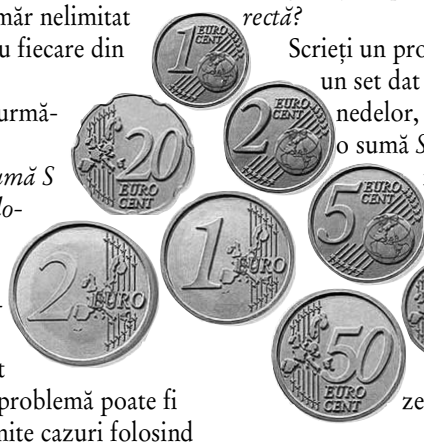
Plovdiv, iulie 2004

Olimpiada BALCANICĂ de Informatică 2004

Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare la cea de-a douăsprezecea ediție a competiției la care se întâlnesc cei mai merituoși elevi informaticieni din Peninsula Balcanică.

P040613: Monede

Se consideră M numere întregi pozitive dintre care unul este 1. Există, de asemenea, un număr nelimitat de monede pentru fiecare din valorile date.



Se consideră următoarea problemă:

O anumită sumă S trebuie plătită folosind un număr minim de monede cu valorile date.

Este cunoscut faptul că această problemă poate fi rezolvată în anumite cazuri folosind următorul algoritm *greedy*:

Se caută moneda cu cea mai mare valoare care este mai mică sau egală cu suma S *după care se scade valoarea monedei găsite din* S. *Se continuă în aceeași manieră până când* S *devine* 0. *Numărul de monede folosit de algoritm pentru a reduce* S *la* 0 *pare să fie numărul minim de monede necesare*.

În multe cazuri algoritmul de mai sus este eficient, dar pentru anumite seturi de valori ale monedelor și anumite valori ale sumei S, algoritmul descris nu duce la obținerea soluției optime.

De exemplu pentru setul de valori {1, 2, 5, 7, 10} și pentru S = 14, algoritmul *greedy* dă o soluție cu trei monede(14 = 10 + 2 + 2) pe cînd soluția minimală este cu două monede (14 = 7 + 7).

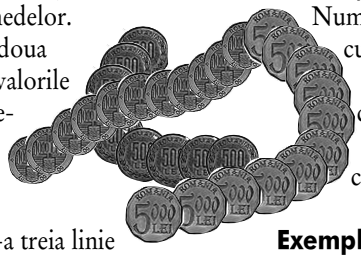
Astfel apare problema: *pentru ce set de valori ale monedelor algoritmul greedy nu produce o soluție corectă?*

Scrieți un program care, pentru un set dat de valori ale monedelor, decide dacă există o sumă S care poate fi obținută folosind un număr de monede mai mic decât cel obținut folosind algoritmul *greedy* prezentat anterior.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține numărul M al valorilor monedelor.



Cea de-a doua linie conține valorile celor M monede, separate prin câte un spațiu.

Pe cea de-a treia linie se află două numere întregi x și y, separate printr-un spațiu. Semnificația acestor valori va fi prezentată în momentul descrierii datelor de ieșire.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Pe prima linie va fi scrisă o valoare S, cuprinsă între valorile x și y specificate la intrare, pentru care algoritmul *greedy* nu duce la obținerea soluției optime.

Cea de-a doua linie trebuie să conțină numerele $b_1, b_2, \ldots, b_M$ care reprezintă numărul de monede de fiecare valoare (în aceeași ordine ca la citire) folosite pentru obținerea sumei S. Numărul total de monede folosite trebuie să fie mai mic decât numărul de monede obținut folosind algoritmul *greedy*.

În cazul în care există mai multe soluții trebuie generată doar una dintre ele.

Se garantează faptul că datele de intrare vor fi astfel alese încât pentru cel puțin una dintre valorile S cuprinse între x și y, algoritmul *greedy* nu va duce la obținerea soluției optime.

Restricții

Numărul tipurilor de monede este cuprins între 1 și 100.

Valorile monedelor sunt cuprinse între 1 și 7.000.000.

Valorile x și y sunt cuprinse între 1 și 7.000.000.

Exemplu

Intrarea standard

```
5
1 2 5 7 10
1 100
```

Ieșirea standard

```
14
0 0 0 2 0
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040614: Echipa

În scurt timp va avea loc *Olimpiada Interpeninsulară de Informatică*, iar liderii echipei *Peninsulei Balcanice* doresc să aleagă cei mai buni concurenți din peninsulă pentru a obține performanțe cât mai bune.

Din fericire echipa poate fi aleasă dintre *N* candidați, care sunt identificați prin numere întregi cuprinse între 1 și *N*.

Pentru a selecta cei mai buni candidați au fost organizate trei competiții.

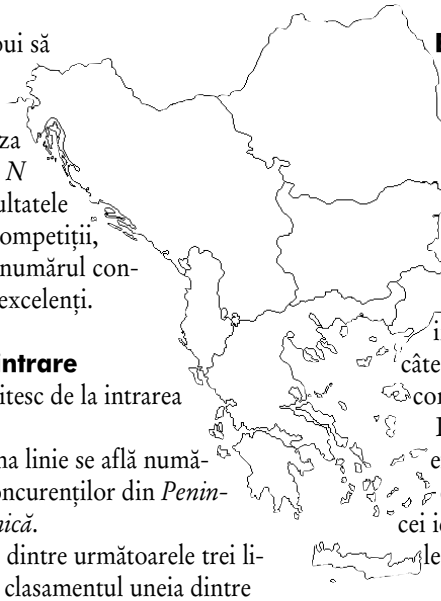
Fiecare dintre cei *N* candidați a participat la toate competițiile, iar la fiecare competiție nu au existat doi sau mai mulți concurenți aflați la egalitate.

Spunem că un concurent *A* este ***mai bun*** decât un concurent *B* dacă *A* s-a clasat înaintea lui *B* în toate competițiile.

Un concurent *A* este excelent dacă nici un concurent nu este mai bun decât *A*.

Pentru a forma cea mai bună echipă, liderii echipei *Peninsulei Balcanice* doresc să afle numărul concurenților excelenți pe care îi au la dispoziție.

Va trebui să scrieți un program care, pe baza unei valori *N* date și rezultatele celor trei competiții, determină numărul concurenților excelenți.



Date de intrare

Datele se citesc de la intrarea standard.

Pe prima linie se află numărul *M* al concurenților din *Peninsula Balcanică*.

Fiecare dintre următoarele trei linii conține clasamentul uneia dintre cele trei competiții care au fost organizate.

O astfel de linie conține numerele de ordine ale concurenților, separate prin câte un spațiu, în ordinea în care s-au clasat aceștia în competiția corespunzătoare liniei.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie care va conține un singur număr, reprezentând numărul concurenților excelenți din *Pensinsula Balcanică*.

Restricție

Numărul concurenților este cuprins între 3 și 500.000.

Exemple

Intrarea standard

```
3
2 3 1
3 1 2
1 2 3
```

Ieșirea standard

```
3
```

Intrarea standard

```
10
2 5 3 8 10 7 1 6 9 4
1 2 3 4 5 6 7 8 9 10
3 8 7 10 5 4 1 2 6 9
```

Ieșirea standard

```
4
```

Explicații

Pentru primul exemplu toți concurenții sunt excelenți deoarece nu există nici un concurent care este mai bun decât altul (pe rând, fiecare concurent s-a aflat înaintea celorlalți doi în câte una dintre cele trei competiții).

Pentru cel de-al doilea exemplu, cei patru concurenți excelenți sunt cei identificați prin numerele 1, 2, 3 și 5.

Timp de execuție: 2.5 secunde/test
Memorie disponibilă: 64 MB

P040615: Drum minim

Compania *Trimoncium Soft* vrea să intre pe piața de *software* cu un produs pentru calculatoare de buzunar cu memorie limitată, care este capabil să presteze o serie de servicii utile via *Internet*.

Un astfel de serviciu oferă utilizatorului posibilitatea de a găsi cea mai scurtă cale între două intersecții ale unui oraș.

Din cauza dificultăților tehnice aplicația ar trebui să facă cât mai puține cereri prin *Internet*.

În oraș se află *N* intersecții, identificate prin numere întregi cuprinse între 1 și *N*.

Poziția unei intersecții *C* este dată de o pereche de coordonate întregi X_C și Y_C într-un sistem de coordonate ortogonal.

Pot exista două sau mai multe intersecții diferite cu aceleași coordonate. Dacă între două astfel de intersecții nu există o stradă, atunci nu se poate trece direct de la o intersecție la cealaltă.

De la fiecare intersecție *C* se poate ajunge la alte M_C intersecții, numite vecini ai intersecției *C*, prin străzi pe care se poate circula într-un singur sens. Fiecare stradă pe care se poate circula în ambele sensuri este descrisă de două străzi pe care se poate circula într-un singur sens (sensurile celor două străzi sunt opuse).

Este posibil ca două străzi să se intersecteze, dar punctul de intersecție să nu fie o intersecție de străzi. De exemplu, ele pot fi plasate pe diferite niveluri (există poduri sau tuneluri).

Lungimea unei străzi este distanța dintre cele două intersecții legate de această stradă.

Scrieți un program care, pentru două intersecții date *S* și *F* determină, folosind subprogramele unei biblioteci, cel mai scurt drum de la intersecția *S* până la intersecția *F*.

Numerele întregi *N*, *S* și *F* sunt obținute prin intermediul celor trei argumente ale subprogramului `start`. Acest subprogram trebuie apelat înaintea oricărui alt subprogram.

Coordonatele X_C și Y_C, precum și numărul M_C al vecinilor unei intersecții *C* pot fi obținute prin intermediul ultimelor trei argumente ale

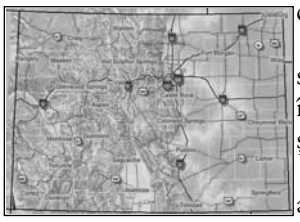
subprogramului `getXYM`. Primul argument al acestui subprogram este valoarea *C*.

Vecinii unei intersecții *C* sunt obținuți prin apelarea subprogramului `getAdj` care are un singur argument: valoarea *C*. Primul apel al acestui subprogram returnează prima intersecție vecină intersecției *C*, al doilea apel returnează a doua intersecție vecină intersecției *C* și așa mai departe până la al M_C-lea apel care returnează cea de-a M_C-a intersecție vecină. Nu este permisă apelarea acestui subprogram decât de cel mult M_C ori, pentru o anumită intersecție *C*.

După ce programul vostru a găsit unul dintre cele mai scurte drumuri de la intersecția *S* până la intersecția *F*, el trebuie să apeleze o singură dată subprogramul `done`, cu un singur argument: numărul *R* al intersecțiilor care se află de-a lungul drumului dintre cele două intersecții (incluzând intersecțiile *S* și *F*).

În continuare, programul trebuie să apeleze subprogramul `report` exact de *R* ori, transmițând de fiecare dată câte un argument: numărul de ordine al unei intersecții (apelurile trebuie efectuate în ordinea în care apar intersecțiile de-a lungul drumului). Ulterior, programul trebuie să se oprească.

După apelarea subprogramului `done`, nu se poate apela nici un alt subprogram cu excepția subprogramului `report`.



Pentru fiecare test programul va fi punctat în funcție de numărul total *K* al apelurilor subprogramelor `getXYM` și `getAdj`.

Acest număr va fi comparat cu numărul *J* de apeluri ale acelorași subprograme de către programul juriului care rezolvă aceeași problemă.

Dacă valoarea *K* este cel mult egală cu valoarea *J*, atunci vor fi acordate 10 puncte pentru testul respectiv.

În caz contrar, vor fi acordate $2 + 4 \cdot (T - K) / (T - J)$ puncte, unde *T* este numărul total de intersecții și străzi.

Dacă programul nu găsește unul dintre cele mai scurte drumuri, dacă încalcă regulile de folosire a bibliotecii, sau dacă răspunde corect accidental fără să asigure corectitudinea (de exemplu, dacă programul definește una din cele mai scurte căi fără nici un apel al subprogramelor `getXYM` și `getAdj`), va fi punctat cu 0 puncte pentru testul respectiv.

Testele sunt formate din date reale, oferite cu bunăvoință de *DATECS GIS Center*.

Biblioteca Pascal

```
procedure start(var N,S,F:
                        LongInt);
procedure getXYM(I: LongInt;
      var XI,YI,MI: LongInt);
function getAdj(I: LongInt):
                         LongInt;
procedure done(R: LongInt);
procedure report(V: LongInt);
```

Biblioteca C/C++

```
void start(long* N, long* S,
                    long* F);
void getXYM(long I, long* XI,
         long* YI, long* MI);
long getAdj(long I);
void done(long R);
void report(long V);
```

Restricții

Numărul intersecțiilor din oraș este cuprins între 10 și 7000.

Coordonatele intersecțiilor sunt numere întregi cuprinse între 0 și 10.000.

Fiecare intersecție are cel mult cinci vecini, iar numărul total al străzilor este de cel mult 16.000.

Experimente

Biblioteca citește harta orașului din fișierul **`path.in`**. Primul rând al acestui fișier va conține numerele întregi *N*, *S* și *F*, separate prin câte un spațiu.

Fiecare dintre următoarele *N* linii conține descrierea unei intersecții, în ordinea numerelor de ordine ale intersecțiilor. Primele trei numere ale liniei vor fi X_C, Y_C și M_C. Ele vor fi urmate de M_C numere care reprezintă numerele de ordine ale intersecțiilor vecine.

În cazul în care programul folosește biblioteca corect și dacă programul returnează un drum valid, atunci drumul va fi descris în fișierul **`path.out`**.

Exemple

`path.in`	**`path.out`**

`3 1 3`
`0 0 1 2`
`0 1 1 3`
`1 1 0`
	`3`
	`1 2 3`

Apel	*Rezultat*
`start(N,S,F)`	`N=3 S=1 F=3`
`getXYM(1,...)`	$X_1 = 0$ $Y_1 = 0$ $M_1 = 1$
`getXYM(3,...)`	$X_3 = 1$ $Y_3 = 1$ $M_3 = 0$
`getAdj(1)`	`2`
`getXYM(2,...)`	$X_2 = 0$ $Y_2 = 1$ $M_2 = 1$
`getAdj(2)`	`3`
`done(3)`	
`report(1)`	
`report(2)`	
`report(3)`	

`path.in`	**`path.out`**

`4 1 4`
`0 0 2 2 3`
`0 1 1 4`
`9 0 2 1 4`
`1 1 1 3`
	`3`
	`1 2 4`

Apel	*Rezultat*
`start(N,S,F)`	`N=4 S=1 F=4`
`getXYM(1,...)`	$X_1 = 0$ $Y_1 = 0$ $M_1 = 2$
`getXYM(4,...)`	$X_4 = 1$ $Y_4 = 1$ $M_4 = 1$
`getAdj(1)`	`2`
`getAdj(1)`	`3`
`getXYM(2,...)`	$X_2 = 0$ $Y_2 = 1$ $M_2 = 1$
`getXYM(3,...)`	$X_3 = 9$ $Y_3 = 0$ $M_3 = 2$
`getAdj(2)`	`4`
`done(3)`	
`report(1)`	
`report(2)`	
`report(4)`	

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

DATECS GIS CENTER

P040616: Cursă

Primarul orașului *Plovdiv* a decis să organizeze o cursă pe străzile orașului pentru a demonstra că acestea sunt potrivite pentru viteze mari. El trebuie să aleagă o rută pentru concurs care să fie cât mai rapidă cu putință.

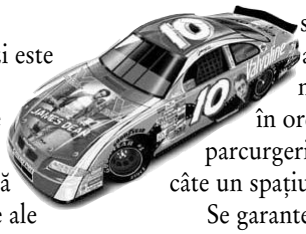
După ce a vorbit cu consilierii săi el a stabilit următoarele restricții. Cursa trebuie să înceapă și să se termine în intersecția unde se află primăria. Singurele schimbări de direcție acceptate de-a lungul traseului sunt cele spre stânga (a face o schimbare de direcție spre stânga față de direcția curentă înseamnă să schimbăm direcția cu un unghi mai mare sau egal cu 0, dar strict mai mic ca 180 de grade).

Dacă există mai multe rute ce îndeplinesc condițiile de mai sus, ruta aleasă va fi aceea pentru care cea mai scurtă stradă a rutei este cât mai lungă posibil.

Orașul *Plovdiv* are N intersecții între care se află un număr total de M străzi pe care se poate circula în ambele sensuri.

Intersecțiile sunt descrise prin două coordonate în plan și sunt numerotate de la 1 la N în ordinea în care sunt date la intrare.

Primăria se află situată în intersecția cu numărul 1. Străzile sunt linii drepte care încep într-o intersecție și se termină în alta.



Lungimea unei străzi este egală cu distanța euclidiană dintre intersecțiile pe care le unește.

O stradă este definită prin numerele de ordine ale intersecțiilor pe care le unește. Nu există mai mult de o stradă între oricare două intersecții ale orașului. Străzile nu se pot intersecta decât la capete.

Scrieți un program care determină o rută, dintre cele posibile în oraș, care începe și se termină cu intersecția 1, iar din fiecare intersecție aflată pe această rută se merge fie înainte, fie se schimbă direcția spre stânga.

Dacă există mai multe astfel de rute, va trebui aleasă cea a cărei cea mai scurtă stradă are lungimea cea mai mare. Ruta aleasă nu poate trece de două ori în același sens pe aceeași stradă.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține două numere întregi N și M, separate printr-un spațiu. Acestea reprezintă numărul intersecțiilor din oraș, respectiv numărul total al străzilor din *Plovdiv*.

Fiecare din următoarele N linii conține coordonatele X și Y ale unei intersecții, separate printr-un spațiu.

Pe ultimele M linii sunt descrise străzile. Fiecare dintre aceste linii conține două numere separate printr-un spațiu, acestea reprezentând numerele de ordine ale intersecțiilor unite de o stradă.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Prima linie va conține numărul de intersecții din rută (intersecția în care se află primăria fiind numărată de două ori).

Următoarea linie va conține numerele de ordine ale intersecțiilor aflate pe ruta aleasă (începând și terminând cu intersecția 1), în ordinea corespunzătoare parcurgerii rutei, separate prin câte un spațiu.

Se garantează faptul că există întotdeauna cel puțin o soluție a problemei.

Dacă există mai multe soluții, poate fi aleasă oricare dintre ele.

Restricții

Numărul intersecțiilor din oraș este cuprins între 3 și 2000.

Numărul străzilor din oraș este cuprins între 5 și 25.000.

Coordonatele intersecțiilor sunt numere întregi cuprinse între -10.000 și 10.000.

Exemplu

Intrarea standard

```
5 6
1 0
2 1
1 1
0 1
1 2
1 2
2 5
1 4
5 4
2 3
4 3
```

Ieșirea standard

```
5
1 2 5 4 1
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040617: Joc

Lui *Ivan* îi place foarte mult să se joace în timpul liber. Din nefericire, el nu se poate bucura întotdeauna de compania prietenilor și câteodată se plictisește fiind singur.

Astfel, *Ivan* inventează jocuri, în care este prezent un singur jucător. El este mândru de ultimul său joc și ar vrea să vă povestească și vouă despre el.

Se dau două șiruri finite de numere întregi pozitive. Jocul constă din mișcări consecutive. Este permisă următoarea mișcare: se șterg ultimele K_1 numere din primul șir (posibil tot șirul) calculându-se suma lor S_1, și ultimele K_2 numere din al doilea șir (posibil tot șirul) calculându-se suma lor S_2. Apoi se determină costul mișcării ca fiind $(S_1 - K_1) \cdot (S_2 - K_2)$.

Mișcările se repetă până la eliminarea tuturor numerelor din ambele șiruri. Costul total al jocului este suma costurilor tuturor mișcărilor. Scopul vostru este de a obține un cost total minim.

Nu este permisă eliminarea tuturor elementelor unui șir, în

timp ce celălalt șir mai conține elemente.

Va trebui să scrieți un program care calculează cel mai mic cost total al jocului prezentat.

Date de intrare
Datele se citesc de la intrarea standard.

Prima linie conține două numere întregi L_1 și L_2, separate printr-un spațiu, care reprezintă lungimile celor două șiruri.

Cea de-a doua linie conține L_1 numere întregi separate prin câte un spațiu, care reprezintă elementele primului șir.

Cea de-a treia linie conține L_2 numere întregi separate prin câte un spațiu, care reprezintă elementele celui de-al doilea șir.

Date de ieșire
Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie pe care se va afla un singur număr care reprezintă costul minim care poate fi obținut pentru jocul descris.

Restricții
La fiecare pas, din fiecare șir va fi șters cel puțin un număr.

Lungimile celor două șiruri sunt cuprinse între 1 și 2000.

Valorile elementelor din cele două șiruri sunt cuprinse între 0 și 1000.

Exemplu
Intrarea standard
```
3 2
1 2 3
1 2
```

Ieșirea standard
```
2
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040618: Coduri
Vom defini distanța $d_H(X, Y)$ dintre două șiruri X și Y de lungimi egale ca fiind distanța *Hamming*. Aceasta reprezintă numărul de poziții în care șirurile X și Y diferă. De exemplu, $d_H(1001, 0010) = 3$ și $d_H(1001111, 0010101) = 4$.

Considerăm un număr întreg strict pozitiv n. Presupunem că $C = W_1, W_2, \ldots, W_M$ este o listă de M șiruri binare de lungime n.

Considerăm că C este o listă circulară și vom defini distanța $d_C(W_i, W_j)$ dintre două șiruri W_i și W_j din listă ca fiind $d_C(W_i, W_j) = \min\{abs(i - j), M - abs(i - j)\}$.

Fie k un număr strict pozitiv mai mic decât n. Vom spune că C este un cod circular de lungime n și anvergură k dacă pentru orice pereche de numere i și j cuprinse între 1 și M sunt respectate fiecare dintre următoarele două condiții:
- dacă $d_C(W_i, W_j) \le k$, atunci $d_H(W_i, W_j) = d_C(W_i, W_j)$;
- dacă $d_C(W_i, W_j) > k$, atunci $d_H(W_i, W_j) > k$.

Problema pricipală în studiul codurilor circulare este determinarea numărului maxim de șiruri dintr-un cod circular de lungime n și anvergură k.

Valoarea exactă a acestui număr este cunoscută doar pentru câteva valori mici ale parameterilor n și k.

Pentru această problemă nu va trebui să creați nici un program, ci să generați doar fișierele de ieșire corecte.

Date de intrare
Urmează să construiți un cod circular conținând cât mai multe șiruri posibile, pentru perechi de valori date ale parametrilor n, k.

Perechile de valori pentru care trebuie determinat codul circular sunt prezentate în figura 1.

Date de ieșire
Trebuie create zece fișiere conținând codurile construite utilizând perechile de parametri din figura 1.

Prima linie a unui fișier trimis trebuie să conțină textul `#FILE code t`, unde t este numărul testului.

Următoarele M linii vor conține șiruri consecutive ale codului de lungime n și anvergură k construit.

Modalitatea de punctare
Pentru fiecare test, cea mai bună dintre toate soluțiile concurenților va primi 10 puncte.

Dacă cea mai bună soluție este un cod format din B șiruri și un concurent trimite o soluție corectă cu M șiruri, atunci punctajul primit va fi de $10 \cdot M / B$ puncte.

Punctajul pentru fiecare test va fi rotunjit astfel încât să conțină cel mult o cifră după virgulă.

Evident, dacă fișierul nu este corect, nu se va acorda nici un punct pentru testul respectiv.

Punctajul total va fi rotunjit la cel mai apropiat întreg.

Exemplu
În cele ce urmează vom considera că pentru testul 0 al problemei avem $n = 4$ și $k = 1$.

O posibilă soluție, pentru care avem $M = 8$, este:

```
#FILE code 0
1101
1100
1110
1010
1011
0011
0001
0101
```

Test #	1	2	3	4	5	6	7	8	9	10
n	5	6	6	7	7	8	8	9	10	10
k	1	1	2	1	2	1	2	2	2	3

Figura 1

Notă
Enunțurile tuturor problemelor prezentate în cadrul acestui număr al revistei au fost traduse de către membrii redacției *GInfo* și au fost modificate pentru a le mări claritatea.