Concursul de Programare „Moisil++”

Concursul de programare „**Moisil++**”**,** desfăşurat la Liceul de Informatică „Grigore C. Moisil” pe data de 13 iunie 2014, s-a adresat elevilor de gimnaziu şi clasa a IX-a, având drept scop dezvoltarea spiritul competitiv şi antrenarea concurenţilor în vederea participării cu succes la alte concursuri de informatică.

Concursul a avut 5 nivele de dificultate, respectiv clasa a V-a, clasa a VI-a, clasa a VII-a, clasa a VIII-a şi clasa a IX-a. Programa de concurs a coincis cu aceea a fazei judeţene a olimpiadei de informatică. La concurs au participat 64 de elevi.

Elaborarea subiectelor şi evaluarea surselor concurenţilor au fost realizate de elevii Marta Filimon, Cezar Andrici, Andrei Netedu, Valentin Roşca, Liana Ţucăr, Mircea Popovici, Francisc Doboş, Adrian Gotcă, Cosmin Pascaru şi studentul Alexandru Cohal sub îndrumarea profesoarelor Cornelia Ivaşc, Mirela Ţibu, Oana Butnăraşu, Liliana Vîrgă, Simona Iuscinschi, Mariana Grădinariu, Silvia Grecu şi Ionel Maftei.

Rezultate

Clasa	Nume	Premiu
5	Ancuta Andrei	I
	Cantea Carmina	I
	Padurariu Robert	I
	Mihalache Mihai	II
	Tescu Tudor	II
	Zamfir Corneliu	M
	Lupu Eduard	M
	Dumitrescu Alex	M
	Iacob Vlad	M
	Nicu Stefan	M
6	Chirita Matei	I
	Cojocaru Diana	I
	Udila Andrei	II
	Croitoriu Dan	II
7	Borza Cristina	I
	Turcuman Vlad	I
	Uliuliuc Serafim	III
	Puricoi Catalin	M
8	Rusu Daniel	I
	Turcuman Horia	II
	Banu Denis	III
	Avram Andrei	M
9	Creanga Daniel	I
	Gherasim Albert	I
	Guraliuc Iulian	I
	Turcu Stefanel	II
	Dascalu Laurentin	II
	Marinoriu Radu	III
	Matei Alexandru	III
	Zaharia Bogdan	III
	Moisiu Horia	M
	Apetri Alexandru	M
	Hriscu Ovidiu	M
	Radeanu Razvan	M
	Mihuuleac Narcis	M
	Babuta Filip	M
	Pascu Vlad	M

Problemele de concurs 2014

Clasa a V-a

Joc – Gotcă Adrian **100 puncte**

Gabriel a început să joace un joc pe calculator. Jocul presupune parcurgerea mai multor nivele. În timp ce juca, el a observat că la terminarea fiecărui nivel obţine cu un punct mai mult decât la nivelul precedent.

Cerinţă

Dându-se `P`, punctajul primului nivel şi `N`, numărul nivelului curent, scrieţi un program care să determine câte puncte a acumulat Gabriel la terminarea nivelului `N`.

Date de intrare

În fişierul `joc.in`, pe prima linie, se află un număr natural `P`, iar pe a doua linie numărul natural `N`, fiecare cu semnificaţia din enunţ.

Date de ieşire

În fişierul `joc.out` pe prima linie se află un număr natural reprezentând suma cerută.

Restricţii

$1 \le p,nr \le 2000000$
$1 \le s \le 2000000$

Exemplu

joc.in	joc.out	Observaţii
500 3	1502	După primul nivel – 500 puncte După al doilea nivel - 1001 puncte După al treilea nivel - 1502 puncte

Timp maxim de execuţie/test: 0.1 secunde
Memorie disponibilă: 2 MB / 1 MB pentru stivă

Soluţie - Gotcă Adrian

O soluţie ar fi cea brută. Pornim de la punctajul P, şi ne oprim la punctajul nivelului curent N, şi anume P+N-1, şi contorizăm de fiecare dată suma.

```
#include<fstream>
using namespace std;
ifstream fin("joc.in");
ofstream fout("joc.out");
int N,P,S,i;
int main() {
    fin>>P>>N;
    for (i=P;i<=P+N-1;i++)
        S+=i;
    fout<<S;
    return 0;
}
```

O altă soluţie ar fi formula pentru sumă de termeni consecutivi: (P+P+N-1)*N/2.

Digroot – Doboş Francisc **100 puncte**

La ora de matematică, Francisc a primit ca sarcină să afle rădăcina digitală a `N` numere naturale. Deşi pare complicat el ştie că `rădăcina digitală` a unui număr se obţine calculând suma cifrelor acestuia, apoi suma cifrelor rezultatului până când se obţine o singură cifră, ca şi în următorul exemplu

`65536 → 6 + 5 + 5 + 3 + 6 = 25 → 2 + 5 = 7.`

Cerinţă

Francisc vă roagă să-l ajutaţi să calculeze radăcinile digitale ale celor `N` numere date și să afișeze suma acestora.

Date de intrare

Fișierul de intrare `digroot.in` conține un număr `N`, urmat pe al doilea rând de `N` numere separate prin câte un spațiu.

Date de ieșire

În fișierul de ieșire `digroot.out` se va afișa un singur număr, reprezentând suma cerută.

Restricţii

$1 \le N \le 10000$
$0 \le a_i \le 2000000000$

Exemple

digroot.in	**digroot.out**	**Explicaţii**
3 13 12 2	9	1+3 → 4 1+2 → 3 2 → 2 4+3+2 = 9
4 532978 2234223 9865 4358	19	5+3+2+9+7+8 → 34 → 3+4 → 7 2+2+3+4+2+2+3 → 18 → 1+8 → 9 9+8+6+5 → 28 → 2+8 → 10 → 1+0 → 1 4+3+5+8 → 20 → 2+0 → 2 7+9+1+2 = 19

Timp de execuţie / test: **0.1** secunde
Memorie disponibilă **4 MB/ 1 MB** pentru stivă

Soluţie – Doboş Francisc

Sol.1 Citim pe rând numerele. Fiecărui numar îi facem suma cifrelor, apoi suma cifrelor rezultatului, până când se obţine o singură cifră. Aceasta se adună la rezultat.

Sol.2 Se observă că rădăcina digitală a numerelor 1,2,..n este o valoare din mulţimea (1,2,..9) şi se repetă din 9 in 9. Astfel se poate utiliza formula digroot(x)=x%9 dacă x%9!=0 si digroot(x)=9 dacă x%9=0. In cazul x=0, digroot(0)=0.

```
#include<fstream>
using namespace std;
int n,x,s=0,i;
ifstream fin("digroot.in");
ofstream fout("digroot.out");
int main()
{
    fin>>n;
    for(i=1;i<=n;i++)
    {
        fin>>x;
        s=s+x%9;
        if(x%9==0&&x>0)
            s=s+9;
    }
    fout<<s;
}
```

Olimpics - Doboş Francisc **100 puncte**

La jocurile olimpice din 2016 de la Rio de Janeiro, participa `n` ţări. Fiecare ţară `i` are un lot format din a_i participanţi, $1 \le i \le n$. Trebuie construite vilele din Satul Olimpic, astfel încât fiecare vilă să găzduiască acelaşi număr de persoane şi, în plus, într-o vilă să fie cazaţi doar participanţi din aceeaşi ţară.

Cerinţă
Ştiind numărul de sportivi din lotul olimpic al fiecarei ţări, determinaţi numărul maxim de sportivi ce vor fi cazaţi în fiecare vilă.

Date de intrare
Din fişierul de intrare olimpics.in se citeşte de pe prima linie un număr N, repezentând numărul de ţări, urmat pe a doua linie de n numere naturale, reprezentând numărul de sportivi din lotul olimpic al fiecărei ţări.

Date de ieşire
În fişierul de ieşire olimpics.out, se va scrie pe prima linie un singur număr reprezentând numărul maxim de sportivi ce vor fi cazaţi în fiecare vilă.

Restricţii
1<=n<=10000
Toate vilele vor fi ocupate în totalitate, nerămânând locuri libere în nicio vilă.

Exemplu

olimpics.in	olimpics.out	Explicaţii
2 12 18	6	Numărul maxim de sportivi din acelaşi lot care vor fi cazaţi într-o vilă este 6.
6 39 130 26 130 260 52	13	Numărul maxim de sportivi din acelaşi lot care vor fi cazaţi într-o vilă este 13.

Timp maxim de execuţie/test: 0.1 secunde Memorie disponibilă: 2 MB / 1 MB pentru stivă

Soluţie – Doboş Francisc

După ce citim toate numerele şi le memorăm într-un vector, parcurgem vectorul, şi facem cmmdc la două câte două elemente consecutive, ţinând minte cmmdc-ul fiecarei perechi într-un alt vector, apoi luăm celălalt vector şi facem cmmdc a două câte două numere, ţinând minte în primul vector rezultatele. Ţinem cont şi de cazul în care sunt un număr impar de elemente, în acest caz punem ultimul element în celalalt vector fără a-l modifica. Repetăm acest proces până ajungem la un singur element, acesta fiind rezultatul.

```
#include<fstream>
using namespace std;
int v1[100001],v2[50001],i,n,j,x,y,c,ok;
int main() {
    ifstream fin("olimpics.in");
    ofstream fout("olimpics.out");
    fin>>n;
    for(i=1; i<=n; i++) {
        fin>>v1[i];
    }
    while(ok==0) {
        j=0;
        for(i=0; i<n/2; i++) {
            x=v1[2*i+1];
            y=v1[2*i+2];
            while(x%y!=0) {
                c=x%y;
                x=y;
                y=c;
            }
            v2[++j]=y;
        }
        if(n%2==1)
            v2[++j]=v1[n];
        if(j==1) {
            fout<<v2[1];
            n=1;
            ok=1;
        }
        n=0;
        if(ok==0) {
            for(i=0; i<j/2; i++) {
                x=v2[i];
                y=v2[i+1];
                while(x%y!=0) {
                    c=x%y;
                    x=y;
                    y=c;
                }
                v1[++n]=y;
            }
            if(j%2==1)
                v1[++n]=v2[j];
            if(n==1) {
                fout<<v1[1];
                n=1;
                ok=1;
            }
        }
    }
}
```

Clasa a VI-a

Hannibal – Andrici Cezar 100 puncte

Hannibal are N cartonaşe, pe fiecare fiind scris un număr natural. Hannibal îşi alege un număr H şi doreşte să afle care este cartonaşul care conţine numărul cu cele mai multe cifre în comun cu H.

Date de intrare

În fişierul hannibal.in, pe prima linie se află numerele N şi H reprezentând numărul de cartonaşe şi numărul ales de Hannibal. Pe a doua linie se află N numere reprezentând numerele de pe cartonaşe.

Date de ieşire

Fişierul hannibal.out conţine o singură linie pe care este scris numărul de pe cartonaşul care are cele mai multe cifre în comun cu numărul H, ales de Hannibal.

Restricţii

$2 \le N \le 10^5$
$0 \le$ cartonas, $H \le 2^{31}$
În caz că există mai multe cartonaşe cu proprietatea cerută, se va afişa numărul cel mai mare.

hannibal.in	**hannibal.out**	**Explicaţie**
5 453532 45 599 4530 2354 6438	2354	Cartonaşul **2354** are cele mai multe cifre în comun cu numărul la care s-a gândit Hannibal.

Timp de execuţie / test: 1 secundă
Memorie disponibilă: 2 MB / 1 MB pe stivă

Soluţie – Andrici Cezar

Pentru fiecare număr de pe cartonaşe, se verifică câte cifre are în comun cu nu<smărul ales de Hannibal, cu ajutorul unor vectori caracteristici.

```
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

int N, H, x, A[11], B[11], carte_de_vizita, maxim = 0;

int main() {

  freopen("hannibal.in", "r", stdin);
  freopen("hannibal.out", "w", stdout);
```

```
  scanf("%d %d\n", &N, &H);

  int h = H;
  while (h) {
    ++A[h%10];
    h /= 10;
  }

  for (; N; --N) {
    memset(B, 0, sizeof(B));
    scanf("%d", &x);
    int X = x;
    while (X) {
        ++B[X%10];
        X /= 10;
    }

    int cifre_in_comun = 0;
    for (int i = 0; i < 10; ++i) {
      cifre_in_comun += min(A[i], B[i]);
      //daca cifra nu se afla in numarul curent si nici in H variabila se
modifica
    }

    if (cifre_in_comun > maxim) {
      maxim = cifre_in_comun;
      carte_de_vizita = x;
    }
    else
        if (cifre_in_comun == maxim && carte_de_vizita < x) {
            carte_de_vizita = x;
        }
  }

  printf("%d\n", carte_de_vizita);

  return 0;
}
```

Cursa – Mircea Popovici **100 puncte**

Ionel s-a înscris la o cursă cu obstacole ce se va desfăşura pe un teren de forma unui caroiaj de dimensiune $n \times n$. Liniile caroiajului au asociate valori în ordine $L_1, L_2, L_3 \ldots L_n$ iar coloanele au asociate valorile $C_1, C_2, C_3 \ldots C_n$. Cu ajutorul lor se determină valorile t_1 şi t_2 asociate fiecărei căsuţe a caroiajului, care se determină astfel $t_1=10*C_j+L_i$ şi $t_2=10*L_i+C_j$.
Fiecare concurent va alerga de-a lungul unui culoar, corespunzător unei coloane distincte a caroiajului. În mod ciudat, organizatorii nu au plasat obstacole în mod egal pentru fiecare concurent, ci au hotărât să lase Soarta, cel mai mare duşman al concurenţilor, să decida care dintre aceştia vor avea o cursă mai uşoară.
Soarta a decis astfel: Într-o căsuţa va fi obstacol dacă cel puţin unul dintre numerele asociate t_1 şi t_2 este prim.
Ionel a vorbit cu Soarta, iar acesta i-a dezvăluit regula. Pentru că îşi doreşte să câştige vă roagă să îi spuneţi care va fi coloana caroiajului cu cele mai puţine obstacole.

Cerinţă

Cunoscandu-se `n` şi numerele asociate liniilor şi coloanelor să se determine coloana cu cele mai puţine obstacole, precum şi numărul de obstacole pe care aceasta le conţine.

Date de intrare

Fişierului de intrare `cursa.in` va avea următoarea structură:

- Pe prima linie va fi un număr natural n, cu semnificaţia din enunţ.
- Pe cea de a doua linie vor fi date în ordine numerele $L_1, L_2, L_3 \ldots L_n$ asociate liniilor.
- Pe cea de a treia linie vor fi date în ordine numerele $C_1, C_2, C_3 \ldots C_n$ asociate coloanelor.

Date de ieşire

Fişierului de ieşire `cursa.out` va avea următoarea structură:

- Pe prima linie se va afla indicele coloanei cautate.
- Pe cea de a doua se va găsi numărul de obstacole de pe aceasta.

Restricţii şi precizări

- $1 \le n, L_i, C_i \le 10$
- Dacă există mai multe răspunsuri se va scrie cel cu numărul de ordine al coloanei cel mai mic.

Exemplu

cursa.in	cursa.out	Explicaţii
10 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9	6 2	

	0	1	2	3	4	5	6	7	8	9
0			02	03		05		07		
1		11		13	41		**61**	17		91
2	02			23						29
3	03	31	23		43	53		37	83	
4		41		43				47		
5	05			53						59
6		61						67		
7	07	71		73	47		**67**			79
8				83						89
9		91	29			59		97	89	

Orice valoare nenulă din tabel reprezintă unul dintre numerele prime t_1 şi t_2 asociate căsuţei respective a caroiajului. Obstacolele sunt plasate în căsuţele caroiajului care conţin aceste valori.

Timp maxim de execuţie/test: 1 secundă
Memorie totală disponibilă: 8 Mb / 4 Mb pentru stivă

Soluţie – Mircea Popovici

Problema se rezolvă prin parcurgerea matricii, generarea celor 2 numere necesare pentru fiecare poziție din matrice și verificarea proprietății acestora. Pentru optimizare, matricea se parcurge pe coloane și nu pe linii, astfel că se poate reține într-un contor numărul de obstacole de pe coloana parcursă, număr ulterior comparat cu un un minim găsit anterior. Pentru a verifica proprietatea de număr prim, se poate folosi clasica verificare a tuturor divizorilor numărului, dar pentru optimizare se poate folosi și ciurul lui Eratostene.

```
#include <fstream>

using namespace std;

ifstream f("cursa.in");
ofstream g("cursa.out");

int n,c[11],l[11],prim[101],mat[11][11];
int i,j;
int main()
{
    f>>n;
    prim[1]=true;
    prim[0]=true;
    for (i=0;i<n;i++)
        f>>l[i];
    for (i=0;i<n;i++)
        f>>c[i];
    for (int x=3;x<=100;x++)
    {
        if (x%2==0)
            prim[x]=true;
        else
        for (i=3;i<x/2;i++)
            if (x%i==0)
                prim[x]=true;
    }
    for (i=0;i<n;i++)
        for (j=0;j<n;j++)
            if (prim[l[i]*10+c[j]]==false || prim[l[i]+10*c[j]]==false)
                mat[i][j]=1;
    int cont,min=11,poz=-1;
    for (j=0;j<n;j++)
    {
        cont=0;
        for (i=0;i<n;i++)
            if (mat[i][j])
                cont++;
        if (cont<min)
            min=cont,poz=j;
    }
    g<<poz<<'\n'<<min<<'\n';
    g.close();
}
```

Olimpiada - Netedu Andrei **100 puncte**

La Olimpiada Naţională de Informatică au participat `n` elevi, Vasile fiind unul dintre ei. El a luat `x` puncte dar nu este mulţumit deoarece el crede că trebuia să obţină un punctaj mai mare. Vasile depune contestaţie şi în urma acesteia, comisia îşi dă seama că trebuie să reevalueze toate sursele concurenţilor. Contestaţia lui Vasile nu a fost întemeiată, în schimb s-au micşorat cele mai mari 3 punctaje diferite cu `p`, `q` şi respectiv `k` puncte.

Cerinţa
Scrieţi un program care să determine pe ce loc era Vasile înainte de contestaţie şi ce loc ocupă el după contestaţie.

Date de intrare
Pe prima linie a fişierului `olimpiada.in` se află `5` numere naturale `n, p, q, k, x` cu semnificaţiile precizate anterior. Pe următoarele `n` linii se află rezultatele obţinute de participanţii de la olimpiadă înainte de contestaţie.

Date de ieşire
Pe prima linie a fişierului `olimpiada.out` se afla locul pe care îl ocupa Vasile înainte de contestaţie, iar pe cea de-a doua linie se află locul pe care îl ocupă el după contestaţie.

Restricţii şi precizări
- `7 < n < 10000`
- `0 < p, q, k, x < 300`
- Punctajul lui Vasile se afla în lista cu cele `n` punctaje.
- Dacă doi concurenţi au acelaşi punctaj atunci ei primesc în clasament acelaşi număr de ordine.

Exemplu

olimpiada.in	olimpiada.out	Explicatii
10 40 20 10 80 99 85 99 80 77 24 11 26 100 200	5 4	În clasamentul iniţial Vasile (80 puncte) era pe locul `5` (punctajele mai mari decât cel al lui Vasile erau: `200, 100, 99, 85)`. Se fac următoarele modificări: `200-40=160; 100-20=80; 99-10=89.` Punctajul lui Vasile rămâne neschimbat, iar noul lui loc în clasament este `4`.

Timp de execuţie / test: 1 secundă
Memorie disponibilă: 2 MB / 1 MB pe stivă

Soluţie - Netedu Andrei şi Pascaru Cosmin

Problema se rezolvă sortând o dată punctajele elevilor şi căutând poziţia lui Vasile. Apoi scădem din primele 3 punctaje valorile respective. Acum mai ordonăm o dată punctajele după modificare şi identificăm poziţia lui Vasile. Trebuie să avem grijă de fiecare dată, pentru că punctajele se pot repeta.

```
#include <fstream>
#include <algorithm>

using namespace std;

ifstream fin("olimpiada.in");
ofstream fout("olimpiada.out");

int v[10005];
int n, p, q, k, x;
```

```
int main()
{
    fin >> n >> p >> q >> k >> x;
    for (int i = 1; i <= n; ++i)
        fin >> v[i];
    sort(v+1, v+1+n);
    int poz = 1;
    int g = n;
    while ( v[g] > x)
    {
        if (v[g]!=v[g-1]) ++poz;
        --g;
    }
    fout << poz << '\n';

    int a = n;
    int b = n;

    do
    {
        if (v[a] == x) x -= p;
        v[a] -= p;
        --a;

    }while(v[a] - p == v[b]);
    b = a;

    do
    {
        if (v[a] == x) x -= q;
        v[a] -= q;
        --a;

    }while(v[a] - q == v[b]);
    b = a;

    do
    {
        if (v[a] == x) x -= k;
        v[a] -= k;
        --a;

    } while(v[a] - k == v[b]);
    b = a;

    poz = 1;
    g = n;
    sort(v+1, v+1+n);
    while ( v[g] > x)
    {
        if (v[g]!=v[g-1]) ++poz;
        --g;
    }
    fout << poz << '\n';

    return 0;
}
```

Clasa a VII-a

Pretenţii - Cohal Alexandru 100 puncte

Zilla a pierdut un pariu cu Alf şi, drept pedeapsă, trebuie să se ducă să îi cumpere acestuia felul său preferat de Kozonak. Zilla îşi notează pe o hârtie ce ingrediente vrea Alf să conţină Kozonakul şi ce ingrediente să nu conţină (pentru ca pedeapsa să fie adevărată, Alf a avut grijă ca lista să fie „lungă să-i ajungă"). Se duce apoi la cea mai apropiată brutărie şi începe să studieze lista cu toate tipurile disponibile de Kozonaki şi ingredientele acestora.
Dar sunt prea multe tipuri de Kozonaki şi lista cu pretenţii a lui Alf este tare lungă, aşadar Zilla are nevoie de ajutor.

Cerinţă

Cunoscând lista cu pretenţii a lui Alf precum, meniul cu toate tipurile de Kozonaki disponibili şi ingredientele acestora, găsiţi numărul de Kozonaki care respectă cerinţele. În plus, deoarece Zilla vrea să cheltuie cel mult **S** Koroane pe Kozonakul lui Alf, trebuie să îi selectaţi doar pe aceia care nu depăşesc această sumă.

Date de intrare

Fişierul de intrare `pretentii.in` are următoarea structură:

- Prima linie conţine numărul natural `S`
- A doua linie conţine numărul natural `n` reprezentând numărul de pretenţii impuse de Alf
- Următoarele `n` linii conţin pretenţiile lui Alf sub forma: `nume_ingredient condiţie` (unde condiţie poate fi **DA** dacă Alf vrea ca respectivul ingredient să apară în Kozonakul său, respectiv **NU** în caz contrar)
- Următoarea linie conţine numărul natural `m` reprezentând numărul de Kozonaki disponibili la brutărie
- Următoarele `m` linii conţin numerele de ingrediente, ingredientele şi preţul fiecărui tip de Kozonak sub forma: `K` $ingredient_1$ $ingredient_2$ ... $ingredient_K$ `pret` (unde `K` reprezintă numărul de ingrediente)

Date de ieşire

Fişierul de ieşire `pretentii.out` va conţine pe prima linie numărul natural `T` reprezentând numărul de tipuri de Kozonaki care respectă condiţiile lui Alf precum şi condiţia lui Zilla.

Restricţii şi precizări

- $1 \le n \le 50$, $1 \le m \le 100$
- 1 ≤ *Numărul de ingrediente* ale unui Kozonak ≤ 65
- 1 ≤ `S`, *Preţul* fiecărui Kozonak ≤ 1000
- *Numele ingredientelor* conţin numai litere mici ale alfabetului latin
- 2≤ Lungimea oricărui ingredient ≤ 20
- În lista cu pretenţii, *Numele unui ingredient* este despărţit de *Condiţia* aferentă de exact un spaţiu
- *Condiţia* poate fi **DA** sau **NU** (aceste două posibile condiţii sunt scrise cu litere mari)
- În lista cu opţiunile disponibile, *Numărul de ingrediente*, *Ingredientele* precum şi *Preţul* unui Kozonak sunt despărţite între ele de exact un spaţiu
- Pentru fiecare Kozonak din meniul disponibil, ingredientele nu se repetă
- Într-un Kozonak selectat toate ingredientele din lista cu pretenţii care au condiţia DA **trebuie** să apară. Orice alt ingredient poate apărea, cu excepţia celor din lista cu pretenţii care au condiţia NU.

Exemplu

pretentii.in	pretentii.out	Explicaţii:
50 6 mere DA cacao NU rosii NU branza DA castraveti DA dovleac NU 5 3 verdeata cacao branzica 20 4 branza ciuperci mere castraveti 40 2 castraveti mere 40 6 ceapa mere cacao castraveti peste branza 60 4 mere castraveti oua branza 80	1	Un singur tip de Kozonak respectă toate cerinţele lui Alf precum şi cerinţa legată de preţ a lui Zilla: Kozonakul al doilea. Ultimul Kozonak respectă toate cerinţele lui Alf însă este prea scump. Celelalte tipuri de Kozonaki nu respectă cerinţele lui Alf.

Timp maxim de execuţie/test: 1 secundă **Memorie totală disponibilă:** 8 Mb / 4 Mb pentru stivă

Soluţie – Cohal Alexandru

Se citesc pretenţiile pe rând şi se reţin în doi vectori ingredientul implicat respectiv condiţia impusă. Pentru simplitate, condiţia se poate memora folosind valorile întregi `0` şi `1` în loc de şiruri de caractere. Pentru o ulterioară simplificare, se memoreaza numărul condiţiilor care impun prezenţa unui anumit ingredient.

Se parcurg opţiunile din meniu pe rând şi se analizează:

- Dacă un Kozonak are mai puţine ingrediente decât numărul de ingrediente impuse să fie prezente atunci sigur opţiunea respectivă **nu** este valabilă şi putem să nu mai facem celelalte verificări pentru a economisi timp
- Dacă se întâlneşte un ingredient care este specificat că nu trebuie să apară atunci opţiunea respectivă sigur **nu** este valabilă
- Pentru fiecare opţiune, se vor număra ingredientele care apar şi care sunt specificate că trebuie să apară. Dacă acest număr este egal cu numărul de ingrediente impuse să apară atunci opţiunea respectivă este o **candidată** să fie valabilă (trebuie îndeplinite şi celelalte restricţii în acelaşi timp)
- Se verifică preţul fiecărui Kozonak să fie mai mic sau cel mult egal cu limita impusă. Dacă această condiţie este îndeplinită atunci opţiunea respectivă este o **candidată** să fie valabilă

Dacă pentru o opţiune se respectă în acelaşi timp toate condiţiile (să fie prezente toate ingredientele impuse să fie prezente, să lipsească toate ingredientele impuse să lipsească, preţul să fie în intervalul dat) atunci avem o opţiune valabilă pe care o contorizăm.

Întrucât timpul de execuţie impus pentru fiecare test nu reprezintă o problemă, pentru fiecare opţiune din meniul brutăriei se va lua fiecare ingredient şi se va căuta în lista cu pretenţii. În funcţie de prezenţa acestui ingredient în lista cu pretenţii (şi eventual condiţia asociată în cazul în care există în lista cu pretenţii) se vor lua deciziile prezentate anterior.

```
#include<fstream>
#include<cstring>
using namespace std;
#define NMAX 100
#define LMAX 20
int s, n, m;
char ingredient_pretentie[NMAX][LMAX];
int conditie_pretentie[NMAX];
int nr_conditie_DA;
int nr_ingrediente_cerute_existente;
int nr_solutii;

int verificare_ingredient(char ingredient[LMAX])
//returneaza 1 daca ingredientul care apare in Kozonakul curent nu este
//conditionat sa nu apara
//returneaza 0 daca ingredientul care apare in Kozonakul curent este
//conditionat sa nu apara
{
	int i;

	for (i = 1; i <= n; ++i)
		if (strcmp(ingredient_pretentie[i], ingredient) == 0)
			if (conditie_pretentie[i] == 1)
			{
				nr_ingrediente_cerute_existente++;
				return 1;
			}
			else
				return 0;

	return 1;
}
```

```
void citire_rezolvare()
{
      int pret;
      int i, j;
      int valid;
      char conditie[3];
      int nr_ingrediente;
      char ingredient[LMAX];

      ifstream fin("pretentii.in");

      fin >> s; //citim suma impusa

      fin >> n; //citim numarul de pretentii

      for (i = 1; i <= n; ++i)
       //pentru fiecare pretentie citim si memoram ingredientul si conditia
      {
                  fin >> ingredient_pretentie[i] >> conditie;
                  if (conditie[0] == 'D')
                  {
                        conditie_pretentie[i] = 1;
                        nr_conditie_DA++;
                         //numaram cate ingrediente sunt impuse sa apara
                  }
                  else
                        conditie_pretentie[i] = 0;
      }

      fin >> m; //citim numarul de pretentii

      for (i=1; i<=m; ++i)
      //pentru fiecare pretentie citim ingredientele si le verificam
      {
            valid = 1;
            nr_ingrediente_cerute_existente = 0;

            fin >> nr_ingrediente;

            if (nr_ingrediente < nr_conditie_DA)
                  //daca numarul de ingrediente este mai mic decat numarul de
ingrediente impuse sa apara este clar ca nu este o optiune valida
                  valid = 0;

            for (j = 1; j <= nr_ingrediente; ++j)
            {
                  fin >> ingredient; //citim un ingredient

                  if (valid == 1)
                        if (verificare_ingredient(ingredient) == 0) //verificam
care este situatia acestui ingredient
                              valid = 0; //daca ingredientul care apare este
conditionat sa nu apara atunci nu avem o optiune valida
            }

            fin >> pret; //citim pretul Kozonakului curent

            if (valid == 1 && nr_ingrediente_cerute_existente == nr_conditie_DA
&& pret <= s)
```

```
//daca avem o optiune valida pana acum (numarul de ingrediente cel putin egal
//cu numarul de conditionari impuse si
//ingredientele care apar nu sunt conditionate sa nu apara), apar toate
//ingrediente impuse sa apara si
//se respecta conditionarea pretului atunci avem o posibila solutie
                nr_solutii++;
     }

     fin.close();
}

void afisare()
{
     ofstream fout("pretentii.out");

     fout << nr_solutii << '\n';
     //afisam solutia problemei

     fout.close();
}

int main()
{
     citire_rezolvare();
     afisare();

     return 0;
}
```

Tabla – Filimon Marta 100 puncte

Ruxandra a descoperit că tabla din clasa ei este magică. Magia sa constă în faptul că dacă o persoană se gandeşte la două dintre numerele scrise pe tablă atunci suma lor va aparea automat pe aceasta. Descoperind superputerea tablei, ea şi Daniela, prietena ei cea mai bună, au inventat un joc care are urmatoarele reguli:

1) La începutul jocului pe tablă sunt scrise numerele `1` si `2`
2) Jucătorul primeşte un număr natural `x`, pe care trebuie să-l obţină folosind superputerea tablei
3) Un număr nou apare pe tablă când jucătorul se gândeşte la două numere deja existente, nu neapărat distincte
4) Odată început jocul, tabla nu se poate şterge iar pe tablă încap cel mult 1000 de numere
5) Jocul se termină când se obţine x sau când se umple tabla

Fetele vă provoacă să vă jucaţi cu ele. Dacă la final pe tablă va fi scris numarul `x` iar acesta va fi format conform regulilor, ele vă vor răsplăti cu `10` puncte.

Cerinţă

Dându-se numărul natural `x`, să se afişeze numerele apărute pe tablă până la obţinerea lui `x` inclusiv.

Date de intrare

Din fişierul text `tabla.in` se citeşte un număr natural `x`, având semnificaţia din enunţ.

Date de ieşire

În fişierul text `tabla.out` se vor afişa numere de pe tablă.

Restricţii şi precizări

- $3 \le x \le 10^{12}$
- Un număr de pe tablă poate fi format prin adunarea unui număr deja existent cu el însuşi.
- Numerele pot fi scrise în orice ordine.
- Orice variantă de răspuns care respectă regulile jocului este corectă.

tabla.in	tabla.out	Explicaţie
4	1 2 4	1 si 2 au fost scrie pe tablă de fete, iar 4 a fost format prin adunarea lui 2 cu el însuţi. O altă variantă de raspuns corecta ar fi putut fi: 1 2 3 4 3=1+2 4=1+3

Timp maxim de execuţie/test: 1 secundă
Memorie totală disponibilă: 8 Mb / 4 Mb pentru stivă

Soluție – Filimon Marta

Problema este una ad-hoc. La baza rezolvării acesteia stă faptul că orice număr poate fi scris ca sumă de puteri distincte ale lui 2. Ne vom forma pe tablă șirul puterilor lui 2 (1, 2, 4, 8,...) până vom găsi un număr mai mare ca x. Apoi folosind o strategie de tip greedy il vom descompune în sumă de numere care sunt puteri ale lui 2, pentru a ști ce numere trebuie să mai formăm pentru a avea pe tabla numărul x.

```
#include<fstream>
#define NMAX 1000

using namespace std;

ifstream f("tabla.in");
ofstream g("tabla.out");

long long n, a[NMAX], mx, i, nr=0, unu=1;


int main()
{
    f>>n;

    a[0]=1;
    for (i=1; (unu<<i)<=n; ++i, ++nr)
        if (((unu<<i)&n)!=0) mx=i;

    for (i=1; i<=mx; ++i) a[i]=unu<<i;

    nr=mx;
    for (i=mx-1; i>=0; --i)
        if ((n&(unu<<i))!=0)
        {
            ++nr;
            a[nr]=a[nr-1]+a[i];
        }

   // g<<nr+1<<"\n";
    for (i=0; i<=nr; ++i)
        g<<a[i]<<" ";
```

```
    f.close();
    g.close();
    return 0;
}
```

Uşi – Cosmin Pascaru **100 puncte**

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Pe planeta `SILI`, uşile au un sistem de deschidere foarte ciudat. Pe fiecare din ele este scrisă o matrice pătratică de dimensiune `n`, urmată de `k` numere. Pentru a deschide o uşă, trebuie introdus un şir de numere. Foarte puţini sunt cei care ştiu care e sensul acestora, majoritatea doar copiindu-le de fiecare dată de pe o foaie. Dumy, din neatenţie, şi-a pierdut foaia pe care avea "cheia" de la propria casă. Cu greu, a reuşit să afle că pentru a deschide uşa trebuie să introducă numerele care se află pe cele `k` poziţii în şirul obţinut prin parcurgerea matricei în spirală, începând cu colţul din stânga-sus, înspre dreapta (în figura alăturată `1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16`).

Cerinţă

Fiind date matricea şi lista de poziţii inscripţionate pe uşă, determinaţi numerele pe care trebuie să le introducă Dumy pentru a deschide uşa.

Date de intrare

De pe prima linie a fişierului de intrare `usi.in` se citesc numerele naturale `n` şi `k`, reprezentând dimensiunea matricei, respectiv numărul de poziţii de pe care trebuie preluate valorile cheii. Pe următoarele `n` linii se gasesc câte `n` numere separate prin spaţiu, reprezentand matricea initială. Pe următoarea linie se află `k` numere $p_1, p_2 \ldots p_k$, reprezentând poziţiile din şirul obţinut prin parcurgerea în spirală a matricii necesare determinării cheii.

Date de ieşire

Fişierul de ieşire `usi.out` va conţine pe prima linie `k` numere separate prin câte un spaţiu $r_1\ r_2 \ldots r_k$, unde r_i este numărul ce se află în matrice pe poziţia p_i, conform parcurgerii acesteia în spirală.

Restricţii şi precizări

- $1 \le n \le 500$
- $1 \le k \le 250.000$
- $1 \le p_i \le n^2$
- Elementele matricei sunt numere naturale întregi cuprinse între `-30000` şi `30000`

Exemplu:

usi.in	usi.out	Explicaţii:
3 5 5 3 1 0 8 6 7 9 2 1 4 2 3 7	5 6 3 1 7	În parcurgerea în spirală, ordinea elementelor este 5 3 1 6 2 9 7 0 8.

Timp maxim de execuţie/test: 1 secunde
Memorie totală disponibilă: 4 Mb /1 Mb pentru stivă

Soluţie - Cosmin Pascaru şi Cohal Alexandru

Soluția 1:
Se parcurge matricea în spirală și se reține într-un vector de lungime n*n, iar apoi se citește lista de poziții și se afișează elementul corespunzător. (a se urmări soluțiile oficiale)

Complexitate: O(n2)

Soluția 2:
Pentru fiecare poziție se caută elementul corespunzător. Se caută secvențial pătratul din care face parte: se compară poziția cu 4*(n-1), apoi cu 4*(n-1 + n-2) etc. Se caută latura din care face parte, iar apoi elementul aflat pe poziția respectivă.

Complexitate: O(k * n)

Soluția 3:
Este de fapt o optimizare adusă soluției 2, în care pătratul din care face parte elementul este căutat binar. Din cauza dimensiunii mici a lui n, diferența de timp de execuție adusă de aceasta optimizare e nesemnificativă.

Complexitate: O(k * log2 n)

Observație: cele trei soluții au timp de executie aproximativ egal.

```
#include<fstream>
using namespace std;
#define NMAX 505
#define KMAX 250005
int n, k;
int a[NMAX][NMAX]; // matricea data
int deplasare_lin, deplasare_col; // valorile care determina deplasarile in
cele patru directii
int desfasurat[NMAX * NMAX]; // contine toate valorile obtinute in urma
parcurgerii in spirala
void schimbare_sens(){ //schimba sensul: Dreapta -> Jos -> Stanga -> Sus -> ...
    if (deplasare_lin == 0 && deplasare_col == 1)
    // daca se deplaseaza la Stanga
    { // atunci se va deplasa in Jos
        deplasare_lin = 1;
        deplasare_col = 0;
    }
    else if (deplasare_lin == 1 && deplasare_col == 0)
    // daca se deplaseaza in Jos
    { // atunci se va deplasa la Stanga
        deplasare_lin = 0;
        deplasare_col = -1;
    }
    else if (deplasare_lin == 0 && deplasare_col == -1)
    // daca se deplaseaza la Dreapta
    { // atunci se va deplasa in Sus
        deplasare_lin = -1;
        deplasare_col = 0;
    }
    else
    { // atunci se va deplasa la Dreapta
        deplasare_lin = 0;
        deplasare_col = 1;
    }
}
```

```
void parcurgere()
{
    int index = 1;
    int lin = 1, col = 1;

    deplasare_lin = 0;
    deplasare_col = 1;

    desfasurat[1] = a[1][1];
    a[1][1] = -1;

    while (index < n * n)
    // cat timp mai avem elemente ale matricii neluate in considerare
    {
        while (a[lin + deplasare_lin][col + deplasare_col] != -1)
        // cat timp ne pute deplasa in directia setata
        {
            index++;
            lin += deplasare_lin; // actualizare indecsi
            col += deplasare_col;

            //adaugarea elementului curent la lista desfasurata a spiralei
            desfasurat[index] = a[lin][col];
            a[lin][col] = -1;
        }
        // cand nu se mai poate deplasa intr-un sens, schimbam sensul
        schimbare_sens();
    }
}

void bordare()
{
    int i, j;

    // bordare coloane
    for (i = 0; i <= n + 1; ++i)
        a[i][0] = a[i][n + 1] = -1;

    // bordare linii
    for (j = 0; j <= n + 1; ++j)
        a[0][j] = a[n + 1][j] = -1;
}

void citire_rezolvare_afisare()
{
    int i, j;
    int poz;

    ifstream fin("usi.in");
    ofstream fout("usi.out");

    fin >> n >> k;

    for (i = 1; i <= n; ++i)
        for (j = 1; j <= n; ++j)
            fin >> a[i][j];

    bordare();
    parcurgere();
```

```
    for (i = 1; i <= k; ++i)
    {
        fin >> poz;
        fout << desfasurat[poz] << " ";
    }
    fout << '\n';

    fin.close();
    fout.close();
}


int main()
{
    citire_rezolvare_afisare(); // 3 in 1 ! :)

    return 0;
}
```

Clasa a VIII-a

Bun – Liana Ţucăr **100 puncte**

Fiind dată o matrice cu `N` linii şi `M` coloane cu numere naturale nenule de cel mult `4` cifre, să se determine latura maximă a unui pătrat „bun''. Un pătrat „bun'' este o submatrice cu `b` linii şi `b` coloane (`b` număr impar), formată din numere prime, cu excepţia elementului din centru, care **nu** este prim.

Date de intrare

Fişierul `bun.in` conţine pe prima linie `2` numere naturale nenule `N` şi `M`, iar pe următoarele `N` linii câte `M` numere separate printr-un spaţiu.

Date de ieşire

Pe prima linie a fişierului `bun.out` se va scrie un singur număr care reprezentă latura maximă a unui pătrat „bun''.

Restrictii şi precizări

- `2 ≤ N,M ≤1000`;
- Elementele matricei sunt numere cuprinse între `1` şi `9999`;
- `1` nu este numar prim;
- Se garantează existenţa unei solţii.

Exemplu

bun.in	bun.out	Explicaţie
7 7 3 7 3 1 10 11 5 11 14 3 15 5 20 17 47 31 31 29 23 24 19 31 37 47 13 17 42 19 2 43 **93** 23 31 19 17 2 13 3 7 11 13 75 11 17 13 17 47 2 21	5	Pătratul căutat este cel cu colţurile în : Stânga-sus `(2,1)`, dreapta-jos `(7,5)` şi cu centrul în `(5,3)`.

Timp maxim de execuţie/test: 1 secundă

Memorie disponibilă: 16 Mb / 4 Mb pentru stivă

Soluţie - Ţucăr Liana

Soluţia este foate asemanatoare soluţiei de tip brut, însă a suferit câteva modificări pentru optimizare precum folosirea **ciurului lui eratostene**:

```
#include<stdio.h>
#define vmax 10010
#define nmax 1050
long d, n, m, i, j, ii, jj, lat, l1, l2, c1, c2, x, rez;
bool npr[vmax], a[nmax][nmax], ok;


void ciur()
{
    npr[1]=1;
    for (d=2;d*d<=vmax;d++)
        if (!npr[d])
            for (j=2;d*j<=vmax;j++)
                npr[d*j]=1;
}

void citire()
{
    scanf("%ld %ld",&n,&m);
    for (i=1;i<=n;i++)
        for (j=1;j<=m;j++)
        {
            scanf("%ld",&x);
            if (npr[x])
                a[i][j]=0;
            else
                a[i][j]=1;
        }
}

void rezolvare()
{
    for (i=1;i<=n;i++)
        for (j=1;j<=m;j++)
            if (a[i][j]==0)
            {
                lat=1;  l1=l2=i;    c1=c2=j;    ok=1;

                while (ok)
                {
                    l1--;   l2++;   c1--;   c2++;
                    for (jj=c1;jj<=c2;jj++)
                        if (a[l1][jj]+a[l2][jj]<2)
                            ok=0;
                    for (ii=l1;ii<=l2;ii++)
                        if (a[ii][c1]+a[ii][c2]<2)
                            ok=0;
                    if (!ok)
                        break;
                    lat+=2;
                }
                if (lat>rez)
                    rez=lat;
            }
    printf("%ld",rez);
}
```

```
int main()
{
    freopen("bun.in","r",stdin);
    freopen("bun.out","w",stdout);
    ciur();
    citire();
    rezolvare();
    return 0;
}
```

Intervale - Roşca Valentin 100 puncte

Numim `interval închis [a,b], a≤b,` mulţimea numerelor naturale cuprinse între `a` şi `b` inclusiv.
Peste mulţimea numerelor de la `1` la `L` se suprapun `N` intervale închise de forma $[a_i,b_i]$, $1 \le a_i \le b_i \le L$. După suprapunere se obţin intervale libere. Un `interval liber` este format dintr-un şir de lungime maximă de numere consecutive din mulţimea `1,2,...L` care nu aparţin niciunuia dintre intervalele date.

Cerinţă

Fiind date 3 numere natural `L`, `P` şi `N` şi apoi `N` intervale, determinaţi numărul de intervale libere de lungime mai mare sau egală cu `P`.

Date de intrare

De pe prima linie a fişierului de intrare `intervale.in` se citesc numerele naturale `L P N`, cu semnificaţiile din enunţ. Pe linia `i+1`, $1 \le i \le N$ se găsesc două numere a_i b_i separate printr-un spaţiu, reprezentand un interval închis.

Date de ieşire

Fişierul de ieşire `intervale.out` va conţine pe prima linie un singur număr `k` reprezentând numărul de intervale libere mai mari sau egale cu `P`.

Restricţii şi precizări

- $2 \le L \le 2.000.000.000$
- Pentru 40% dintre teste $2 \le L \le 2.000.000$
- Pentru 70% dintre teste $2 \le L \le 16.000.000$
- $1 \le P \le L$
- $1 \le N \le 300.000$

Exemplu:

intervale.in	intervale.out	Explicaţii:
20 4 3 1 3 8 10 16 19	2	Intervalele libere sunt: (4,5,6,7), (11,12,13,14,15) si (20). Primul conţine 4 numere, al doilea 5 numere iar ultimul conţine un singur număr, deci există două intervale cu lungimea mai mare sau egală cu P = 4.

Timp maxim de execuţie/test: 1 secundă
Memorie totală disponibilă: 3 Mb /1 Mb pentru stivă

Soluţie – Roşca Valentin

Scopul problemei este de a afla intervalele libere, dintre care număram numai pe acelea care au lungimea mai mare sau egală ca `P`.

O prima soluţie ar fi să folosim un **vector de frecvenţă** şi să "vizităm" numerele acoperite de intervalele închise, apoi construind intervalele libere pe baza acestuia. Aceasta soluţie obţine `40` de puncte în cazul în care vectorul de frecvenţă este de tip char de mărimea maxim `2.000.000` pentru a nu ieţi din memoria impusă.

O a doua soluţie, fiind o optimizare a memoriei primei soluţii , se bazează pe faptul că tipul `char` are mărimea de **`1 byte = 8 biţi`**. Deoarece vectorul de frecventa nu are nevoie decat sa reţina `0` sau `1` pentru fiecare element, vom memora vectorul de frecvenţă pe biţi, memoria necesară micşorându-se de 8 ori. Aceasta soluţie obţine `70` de puncte.

În soluţia de `100` de puncte folosim metoda **greedy**, unde reţinem într-un vector doar capetele intervalelor.

Un interval liber reprezintă numerele dintre două intervale închise, distjuncte şi consecutive.

Pentru a afla intervalele consecutive este necesar ca mai întai vectorul să fie sortat după capătul din stanga(cel mai mic dintre cele două) al fiecărui interval . Apoi pentru fiecare interval închis, **de la primul interval în ordine**, calculam distanţa de la intervalul curent şi intervalul precedent, aceasta fiind distanţa intervalului liber dintre ele. În cazul în care capătul din stanga al intervalului current este mai mic sau egal cu capătul din dreapta al intervalului precedent(au elemente comune), distanţa rezultată va fi mai mică sau egală cu `0`.

```
#include <fstream>
#include <string.h>

using namespace std;

struct interval {
    int a, b;
};
interval v[300001];

int main ( ) {
    ifstream fin ("intervale.in");
    ofstream fout ("intervale.out");
    int n, l, p, st, nr = 0, a, b;
    fin >> l >> p >> n;
    for ( long i = 1; i <= n; i++ ) {
        fin >> a >> b;
        v[i].a = a; v[i].b = b;
    }

    fin.close ();

    //sortam intervalele prin metoda bubble-sort
    for (int i = 1; i < n; i++)
        for ( int j = i + 1; j <= n; j++ )
            if ( v[i].a > v[j]. a || (v[i].a == v[j].a) && v[i].b > v[j].b ) {
                interval aux = v[i];
                v[i] = v[j];
                v[j] = aux;
            }
```

```
    st = 1; // primul element din intervalul liber
    int i = 1;
    while ( i <= n ) {
        if ( st < v[i].a ) {
            //cazul in care exista un interval liber
            if ( v[i].a - st >= p )
                nr++;

            //primul element din intervalul liber se schimba
            st = v[i].b + 1;
        }
        else
            if ( v[i].a <= st && st <= v[i].b )
  //cazul in care intervalul precedent are elemente comune cu intervalul curent
                st = v[i].b + 1;
        i++;
    }

    if ( st <= l )
        if ( l - st + 1 >= p )
            nr++;

    fout << nr;
    return 0;
}
```

Navete – Cohal Alexandru **100 puncte**

Pe planeta Kubus unde trăiesc Alf şi Zilla nu există maşini, există doar navete zburătoare. Acestea, la fel ca şi maşinile de pe Pământ, au plăcuţe de înmatriculare prin care navetele sunt identificate **în mod unic**. Codul de pe o astfel de plăcuţă a unei maşini are următoarul format **`LLLccnnS`**, unde:

- **`LLL` – 3 litere mari ale alfabetului latin** (`'A'... 'Z'`) preluate din numele dinozaurului persoanei care deţine naveta despre care vorbim (fiecare locuitor de pe Kubus are un dinozaur ca animal de companie);
- **`cc` – 2 caractere cifră** (`'0'... '9'`) care reprezintă numărul de prieteni imaginari pe care i-a avut proprietarul maşinii în copilărie (ex: `"02"`– `2` prieteni; `"32"` – `32` prieteni);
- **`nn` – 2 litere mici ale alfabetului latin** (`'a'... 'z'`) reprezentând primele două litere din *numele de telefon* al proprietarului (telefoanele locuitorilor de pe Kubus sunt identificate prin nume şi nu prin numere ca pe Pământ; daca cineva vrea să îl sune pe proprietarul unei maşini trebuie să ştie de unde să înceapă căutarea);
- **`S` – 1 caracter special** (`'#', '&', '%', '?', '!', '@', '$'`) care semnifică codul mâncării preferate a proprietarului maşinii (pe Kubus există doar `7` feluri de mâncare, fără carne de dinozaur).

Într-o zi, Alf şi Zilla au stat la marginea unui culoar de zbor al navetelor de pe planeta Kubus şi au notat `n` coduri ale plăcuţelor de înmatriculare ale navetelor care au zburat prin dreptul lor.

Cerinţă

Alf şi Zilla vor să ştie:

a) câte dintre cele `n` navete zburătoare observate au fost înmatriculate pe planeta Kubus, indiferent dacă sunt distincte sau nu (o navetă înmatriculată pe altă planetă are alt format al codului);
b) care este cea mai des întâlnită literă dintre primele trei litere din numelor dinozaurilor, considerând doar navetele înmatriculate pe Kubus, indiferent dacă sunt distincte sau nu;(dacă există mai multe litere care apar de acelaşi număr de ori în numele dinozaurilor, se va alege litera cea mai mică în ordine lexicografică)
c) care este codul navetei care a trecut de cele mai multe ori prin dreptul lor;(dacă există mai multe navete care apar de acelaşi număr de ori în lista celor doi, se va alege naveta cu codul cel mai mic lexicografic);
d) numărul total al prietenilor imaginari avuţi de proprietarii de navete, considerând doar navetele distincte înmatriculate pe Kubus.

Date de intrare

Pe prima linie a fişierului `navete.in` se află numărul natural `n`. Pe următoarele `n` linii se vor afla codurile de pe plăcuţele de înmatriculare notate de Alf şi de Zilla.

Date de ieşire

Fişierul de ieşire `navete.out` conţine patru linii, fiecare linie conţinând răspunsul la cerinţele formulate, în ordinea precizată.

Restricţii

- $1 \le n \le 300$
- Fiecare cod de înmatriculare va avea lungimea din intervalul `[1,30]` şi va conţine doar `litere mari` şi `mici` din alfabetul latin, `cifre` şi `caractere speciale`('#','&','%','?','!','@','$')
- Dintre cele n navete observate de către cei doi, cel puţin una este înmatriculată pe Kubus
- Se vor acorda punctaje parţiale astfel: primele două cerinţe valorează fiecare câte `20%` din punctajul testului, iar ultimele două cerinţe valorează fiecare câte `30%` din punctajul testului.

Exemplu

navete.in	**navete.out**	**Explicaţie**
7 SMP23ty& TMK02pt& ERG876ut@ TMK02pt& AFT00ra? MOISIL++ FBF00as%	5 F TMK02pt& 25	A treia navetă nu respectă formatul unui cod de înmatriculare deoarece are lungimea de `9` caractere, iar pe planeta Kubus codurile de înmatriculare conţin `8` caractere. A şasea navetă nu respetă formatul codurilor de înmatriculare deoarece numărul de prieteni imaginari ai proprietarului este format din litere şi nu din cifre. Literele `F`, `M` şi `T` se întâlnesc de cele mai multe ori (de trei ori), dintre acestea alegem `F` deoarece este mai mică din punct de vedere lexicografic. Naveta `TMK02pt&` este singura care a trecut mai mult de o dată. `23 + 2 + 0 + 0 = 25` prieteni imaginari.

Timp maxim de execuţie/test: 0.1 secunde **Memorie disponibilă:** 16 Mb / 8 Mb pentru stivă

Soluţie – Cohal Alexandru

Se citesc cele `n` coduri. Se verifică care dintre acestea respectă formatul precizat și sunt alcătuite din exact `8` caractere. Numărul codurilor care verifică aceste condiții reprezintă **soluția primei cerințe**.

Pentru a rezolva a doua cerință vom forma un *vector de frecvență* în care vom reține numărul de apariții ale fiecărei litere dintre primele `3` caractere din fiecare cod corespunzător unei navete înmatriculate pe Kubus. Litera care are frecvența cea mai mare (dacă există mai multe astfel de litere o alegem pe cea care este cea mai mică în ordine lexicografică) reprezintă **soluția celei de-a doua cerințe**.

Pentru a rezolva a treia cerință vom proceda similar. Vom reține toate codurile distincte într-un vector de șiruri de caractere și numărul de apariții corespunzător pentru fiecare cod. Atunci când citim un cod nou, verificăm dacă îl avem deja reținut în vector. Dacă da atunci mărim cu o unitate numărul de apariții. Dacă nu, introducem codul în vector și inițializăm numărul de apariții cu valoarea `1`. Codul care are frecvența (numărul de apariții) cea mai mare (dacă există mai multe astfel de coduri îl alegem pe cel care este cel mai mic în ordine lexicografică) reprezintă **soluția celei de-a treia cerințe**.

Pentru a rezolva ultima cerință, vom trasfoma cele două caractere de pe pozițiile `4` și `5` (numerotând cu `1` prima poziție) din fiecare cod al unei navete înmatriculate pe Kubus în cifre și le vom aduna în mod corespunzător la **soluția celei de-a patra cerințe**.

```
#include<fstream>
#include<string.h>

using namespace std;

ifstream fin("navete.in");
ofstream fout("navete.out");

int n; //numarul total de navete
char s[40]; //codul curent citit
int nrInmatriculatK; //numarul navetelor inmatriculate pe Kubus
int frecventaLitera[260]; //vectorul de frecvente pentru literele mari din nume
int  nrNaveteDistincte,  frecventaNaveteDistincte[310];  //numarul  de  navete
distincte si vectorul lor de frecventa
char naveteDistincte[310][40]; //navetele distincte
int numarPrieteni; //numarul de prieteni imaginari
char literaFrecventaMax; //litera cu frecventa maxima
int navetaFrecventaMax; //numarul navetei cu frecventa maxima

int litereMari()
{
     if (s[0] >= 'A' && s[0] <= 'Z' && s[1] >= 'A' && s[1] <= 'Z' && s[2] >=
'A' && s[2] <= 'Z')
          return 1;
     return 0;
}

int cifre()
{
     if (s[3] >= '0' && s[3] <= '9' && s[4] >= '0' && s[4] <= '9')
          return 1;
     return 0;
}

int litereMici()
{
     if (s[5] >= 'a' && s[5] <= 'z' && s[6] >= 'a' && s[6] <= 'z')
          return 1;
     return 0;
}

int caracterSpecial()
{
     if (s[7] == '#' || s[7] == '&' || s[7] == '%' || s[7] == '?' || s[7] ==
'!' || s[7] == '@' || s[7] == '$')
          return 1;
     return 0;
}

void citire_rezolvare()
{
     int i, j, gasit;

     fin >> n;
     fin.getline(s, 40);

     for (i=1; i<=n; ++i)
     {
          fin.getline(s, 40);

          //verific daca formatul codului este cel specificat
```

```
          if (strlen(s) == 8 && litereMari() && cifre() && litereMici() &&
caracterSpecial())
          {
               //cresc numarul navetelor inmatriculate pe Kubus
               nrInmatriculatK++;

               //numar aparitiile celor trei litere din numele dinozaurilor
               frecventaLitera[s[0]]++;
               frecventaLitera[s[1]]++;
               frecventaLitera[s[2]]++;

               //caut daca a mai trecut naveta curenta anterior
               gasit = 0;
               for (j=1; j<=nrNaveteDistincte; ++j)
                    if (strcmp(s, naveteDistincte[j]) == 0)
                    {
                         frecventaNaveteDistincte[j]++;
                         gasit = 1;
                         break;
                    }

               //daca nu a mai trecut anterior, memorez codul navetei
               if (gasit == 0)
               {
                    nrNaveteDistincte++;
                    strcpy(naveteDistincte[nrNaveteDistincte], s);
                    frecventaNaveteDistincte[nrNaveteDistincte] = 1;

                    //numar prietenii imaginari
                    numarPrieteni += (s[3] - '0') * 10 + (s[4] - '0');
               }
          }
     }
     fin.close();
}

void rezolvare()
{
     int i, frecventaMax;

     //determin litera cu frecventa maxima
     frecventaMax = 0;
     for (i='A'; i<='Z'; ++i)
          if (frecventaLitera[i] > frecventaMax || (frecventaLitera[i] ==
frecventaMax && i < literaFrecventaMax))
          {
               frecventaMax = frecventaLitera[i];
               literaFrecventaMax = i;
          }

     //determin naveta cu frecventa maxima
     frecventaMax = 0;
     for (i=1; i<=nrNaveteDistincte; ++i)
          if (frecventaNaveteDistincte[i] > frecventaMax ||
(frecventaNaveteDistincte[i] == frecventaMax && strcmp(naveteDistincte[i],
naveteDistincte[navetaFrecventaMax]) < 0))
          {
               frecventaMax = frecventaNaveteDistincte[i];
               navetaFrecventaMax = i;
          }
}
```

```
void afisare() {
      fout << nrInmatriculatK << '\n' << literaFrecventaMax << '\n';
      fout << naveteDistincte[navetaFrecventaMax] << '\n';
      fout << numarPrieteni << '\n';
}
int main() {
      citire_rezolvare();
      rezolvare();
      afisare();
      return 0;
}
```

Clasa a IX-a

Seif - Doboş Francisc **100 puncte**

În primul război mondial s-a găsit într-un bunker german un seif plin cu documente strict secrete, inscripţionat cu `N` numere. Aliaţii şi-au trimis cei mai buni criptologi să decodeze cifrul, şi au aflat că este format după regula următoare: fiecărui număr îi este făcută suma cifrelor, şi dacă rezultatul are mai mult de o cifră, procesul se repetă, după exemplul următor:

`1564634 → 1 + 5 + 6 + 4 + 6 + 3 + 4 = 29 → 2 + 9 = 11 → 1 + 1 → 2`

Cifrul se obţine făcând suma celor `N` cifre astfel obţinute.

Cerinţă
Deoarece sunt foarte multe numere, criptologilor le-ar lua prea mult timp să îl decodeze, aşa că iţi cer ţie să scrii un program care să-i ajute.

Date de intrare
În fişierul `seif.in`, pe prima linie, se află un număr natural `N`, iar pe a doua linie `N` numere naturale, fiecare cu semnificaţia din enunţ.

Date de ieşire
În fişierul `seif.out` pe prima linie se află un număr natural reprezentând suma cerută.

Restricţii
$1 \le N \le 10000$
$0 \le a_i \le 2000000000$

Exemplu

seif.in	seif.out	Explicaţii
3 113 47 8	15	1+1+3 → 5 4+7 → 11 → 1+1 → 2 8 → 8 5+2+8 = 15
5 9456 288536 1399 1337 6556	24	9+4+5+6 → 24 → 2+4 → 6 2+8+8+5+3+6 → 32 → 3+2 → 5 1+3+9+9 → 22 → 2+2 → 4 1+3+3+7 → 14 → 1+4 → 5 6+5+5+6 → 22 → 2+2 → 4 6+5+4+5+4 = 24

Timp maxim de execuţie/test: 0.1 secunde
Memorie disponibilă: 4 MB / 1 MB pentru stivă

Soluţie – Doboş Francisc

Sol.1 Citim pe rând numerele. Fiecărui număr îi facem suma cifrelor, apoi suma cifrelor rezultatului, până când se obține o singură cifră. Aceasta se adună la rezultat.
Sol.2 Se observă că encriptarea numerelor 1,2,..n este o valoare din mulțimea (1,2,..9) și se repetă din 9 în 9. Astfel se poate utiliza formula encrypt(x)=x%9 dacă x%9!=0 și encrypt(x)=9 dacă x%9=0. În cazul x=0, encrypt(0)=0.

```
#include<fstream>
using namespace std;
int n,x,s=0,i;
ifstream fin("seif.in");
ofstream fout("seif.out");
int main() {
    fin>>n;
    for(i=1;i<=n;i++) {
        fin>>x;
        s=s+x%9;
        if(x%9==0&&x>0)
            s=s+9;
    }
    fout<<s;
}
```

Turnuri – Doboş Francisc **100 puncte**

În primul război mondial, ca să vadă din timp când sunt atacaţi, Aliaţii au hotărât să construiască turnuri de observaţie. Muncitorii au primit `N` grămezi de blocuri de piatră de formă cubică, de aceleaşi dimensiuni. Prima grămadă conţine blocuri de culoarea `1`, a doua grămadă conţine blocuri de culoarea `2` etc. Muncitorii au primit ordin ca turnurile să aibă înălţimi egale şi fiecare turn să fie făcut din blocuri de aceeaşi culoare. Pentru a se putea vedea cât mai bine, turnurile trebuie să aibă înălţimea maximă posibilă.

Cerinţă
Ştiind numărul de blocuri din fiecare grămadă, determinaţi înălţimea maximă a turnurilor.

Date de intrare
Din fişierul de intrare `turnuri.in` se citeşte de pe prima linie un număr `N`, reprezentând numărul de grămezi, urmat pe a doua linie de `N` numere naturale, reprezentând numărul de blocuri din fiecare grămadă.

Date de ieşire
În fişierul de ieşire `turnuri.out`, se va scrie pe prima linie un singur număr reprezentând înălţimea maximă pe care o pot avea turnurile.

Restricţii
`1≤N≤10000`
Muncitorii trebuie să folosească toate blocurile.

Exemplu

turnuri.in	turnuri.out	Explicaţii
2 12 18	6	Se pot construi blocuri de înălţime maximă 6.
6 39 130 26 130 260 52	13	Se pot construi blocuri de înălţime maximă 13.

Timp maxim de execuţie/test: 0.1 secunde
Memorie disponibilă: 2 MB / 1 MB pentru stivă

Soluţie - Doboş Francisc

După ce citim toate numerele şi le memorăm într-un vector, parcurgem vectorul, şi facem cmmdc la două câte două elemente consecutive, ţinând minte cmmdc-ul fiecarei perechi într-un alt vector, apoi luăm celălalt vector şi facem cmmdc a două câte două numere, ţinând minte în primul vector rezultatele. Ţinem cont şi de cazul în care sunt un număr impar de elemente, în acest caz punem ultimul element în celalalt vector fără a-l modifica. Repetăm acest proces până ajungem la un singur element, acesta fiind rezultatul.

```
#include<fstream>
using namespace std;
int v1[100001],v2[50001],i,n,j,x,y,c,ok;
int main() {
    ifstream fin("turnuri.in");
    ofstream fout("turnuri.out");
    fin>>n;
    for(i=1; i<=n; i++) {
        fin>>v1[i];
    }
    while(ok==0) {
        j=0;
        for(i=0; i<n/2; i++) {
            x=v1[2*i+1];
            y=v1[2*i+2];
            while(x%y!=0) {
                c=x%y;
                x=y;
                y=c;
            }
            v2[++j]=y;
        }
        if(n%2==1)
            v2[++j]=v1[n];
        if(j==1) {
            fout<<v2[1];
            n=1;
            ok=1;
        }
        n=0;
        if(ok==0) {
            for(i=0; i<j/2; i++) {
                x=v2[i];
                y=v2[i+1];
                while(x%y!=0) {
                    c=x%y;
                    x=y;
                    y=c;
                }
                v1[++n]=y;
            }
            if(j%2==1)
                v1[++n]=v2[j];
            if(n==1) {
                fout<<v1[1];
                n=1;
                ok=1;
            }
        }
    }
}
```

Batalion – Roşca Valentin **100 puncte**

Într-o zi, în timpul primului război mondial, cei N soldaţi ai unui batalion au fost aliniaţi în linie dreaptă şi numerotaţi de la 1 la N, pentru a-şi primi ordinele. Ei primesc ordine sub forma unor perechi de numere, de forma a b, cu semnificaţia ca soldaţii numerotaţi consecutiv de la a la b primesc misiunea respectivă. Deşi unii soldaţi primesc mai multe misiuni deodată, rămân câţiva soldaţi fără nicio misiune. Generalul batalionului a hotărât să trimită soldaţii care au rămas fără misiune să păzească puncte cheie ale frontului, dar nu a putut trimite mai puţin de P soldaţi la un astfel de punct, deoarece ar fi fost prea periculos. Pentru a avea şanse de reuşită cât mai mari, a trimis la fiecare punct cheie un şir de lungime cât mai mare de soldaţi numerotaţi consecutiv, care nu au primit anterior nicio misiune.

Cerinţă

Fiind date N−numărul de soldaţi, P−numărul minim de soldaţi ce pot păzi un punct cheie şi M−numărul de misiuni, să se afle numărul de puncte ce pot fi păzite de soldaţi.

Date de intrare

De pe prima linie a fişierului de intrare batalion.in se citesc numerele naturale N P M, cu semnificaţiile din enunţ. Pe linia i+1, $1 \le i \le N$ se găsesc două numere a_i b_i separate printr-un spaţiu, reprezentând soldaţii trimişi în misiunea i.

Date de ieşire

Fişierul de ieşire batalion.out va conţine pe prima linie un singur număr k reprezentând numărul de puncte cheie ce pot fi păzite.

Restricţii

- $2 \le N \le 2.000.000.000$
- Pentru 40% dintre teste $2 \le N \le 2.000.000$
- Pentru 70% dintre teste $2 \le N \le 16.000.000$
- $1 \le P \le N$
- $1 \le M \le 300.000$

Exemplu

batalion.in	batalion.out	Explicaţie
20 5 3 1 2 8 10 17 19	2	Soldaţii rămaşi fără misiune sunt 3 4 5 6 7 11 12 13 14 15 20. Rămân două şiruri de soldaţi numerotaţi consecutiv, cu lungime cel puţin P, care pot să păzească un punct cheie, şi anume 3 4 5 6 7 si 11 12 13 14 15 16.

Timp de execuţie / test: 1 secundă
Memorie disponibilă: 3 MB / 1 MB pe stivă

Soluţie – Roşca Valentin

Scopul problemei este de a afla şirurile libere de soldaţi, dintre care număram numai pe acelea care au lungimea mai mare sau egală ca `P`. Problema se reduce la ordonarea soldaţilor care se afla în misiune şi procesarea acestora în funcţie de soldaţii precedenţi:

```
#include <fstream>
#include <string.h>
using namespace std;
struct soldati {
    int a, b;
};
soldati v[300001];
int main ( ) {
    ifstream fin ("batalion.in");
    ofstream fout ("batalion.out");

    int n, l, p, st, nr = 0, a, b;

    fin >> l >> p >> n;
    for ( long i = 1; i <= n; i++ ) {
        fin >> a >> b;
        v[i].a = a; v[i].b = b; //memoram grupul de soldati aflati in misiune
    }
    fin.close ();

    //sortam grupurile de soldati prin metoda bubble-sort
    for (int i = 1; i < n; i++)
        for ( int j = i + 1; j <= n; j++ )
            if ( v[i].a > v[j]. a || (v[i].a == v[j].a) && v[i].b > v[j].b ) {
                interval aux = v[i];
                v[i] = v[j];
                v[j] = aux;
            }

    st = 1; // primul soldat liber
    int i = 1;
    while ( i <= n ) {
        if ( st < v[i].a ) {
            //cazul in care exista un cel putin un soldat liber
            if ( v[i].a - st >= p )
                nr++;

            //primul soldat liber se schimba
            st = v[i].b + 1;
        }
        else
            if ( v[i].a <= st && st <= v[i].b )
                //cazul in care se repeta soldatii aflati in misiune
                st = v[i].b + 1;
        i++;
    }

    if ( st <= l )
        if ( l - st + 1 >= p )
            nr++;

    fout << nr;
    return 0;
}
```

Problemele de concurs 2013

Clasa a V-a

Şoricei – Luca Roxana **100 puncte**

Trei pisici au ieşit la vânătoare de şoricei. Ele au vânat `n` şoricei, i-au etichetat cu câte un număr natural, în ordinea în care au fost prinşi. Ca să nu se certe între ele de la împărţirea şoriceilor, au hotărât să-i distribuie după următoarele reguli:

- Prima pisică va lua acei şoricei care au etichete cu numere pare
- A doua pisică va lua dintre şoriceii rămaşi pe cei pentru care suma cifrelor numărului de pe etichetă este divizibila cu `13`
- A treia pisică va lua toţi şoriceii rămaşi după ce primele două şi-au primit partea.

Cerinţă

Dându-se numărul şoriceilor şi etichetele lor, să se determine câţi şoricei va mânca fiecare pisică.

Date de intrare

În fişierul de intrare `soricei.in` se va afla pe prima linie numărul natural `n`, numărul de şoricei, iar pe următoarea linie `n` numere naturale reprezentând numerele de pe etichetele şoriceilor.

Date de ieşire

În fişierul `soricei.out` se va afişa pe prima linie `n1`, numărul de şoricei pe care îi va mânca prima pisică, pe a doua linie se va afişa `n2`, numărul de şoricei pe care îi va mânca a doua pisică, pe a treia linie se va afişa `n3`, numărul de şoricei pe care îi va mânca a treia pisică.

Restricţii şi precizări

- `1 ≤ n ≤ 1000`
- Numerele de pe etichetele şoriceilor vor fi cuprinse între `1` şi `10000`.
- **În cazul în care nu ştiţi să aflaţi numărul de şoricei mâncaţi de o pisică, vă rugăm să afişaţi `0` pe linia corespunzătoare ei.**
- Se vor acorda puctaje parţiale astfel: pentru primele două rezultate corecte câte `40%` din punctajul testului şi `20%` pentru ultimul rezultat corect.

soricei.in	**soricei.out**	**Explicaţii**
3 58 841 3	1 1 1	Fiecare pisică manâncă un singur şoricel: • Prima pisici mănâncă şoricelul cu eticheta 58, deoarece este singurul şoricel cu etichetă pară • A doua pisică mănâncă soricelul cu eticheta `841` deoarece este singurul număr rămas pentru care suma cifrelor etichetei este divizibilă cu `13`. • Celei de-a treia pisici îi rămane şoricelul cu eticheta `3`.
7 2 841 3 26 5413 4 7	3 2 2	

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 4Mb / 2 Mb pentru stivă

Soluţie – Luca Roxana

Pentru fiecare pisică vom avea asociată o variabilă care contorizează numărul de șoricei care îi revin.
Vom parcurge etichetele soriceilor. Pentru o eticheta x, aflată pe poziția i în fișierul de intrare, vom testa:

1) Dacă x este un număr par, numărul de șoricei care îi revin primei pisici este mărit cu unu.
2) Daca **suma cifrelor** lui x este **divizibilă cu 13** și nu îndeplinește condiția 1) atunci numărul de șoricei care îi revin celei de a doua pisici este mărit cu unu.
3) Dacă nu îndeplinește nicio condiție de mai sus atunci numărul de șoricei care îi ultimii pisici este mărit cu unu.

```
#include <fstream>

using namespace std;
```

```
    ifstream cin("soricei.in");
    ofstream cout("soricei.out");

    int a=0, b=0, c=0, n, sum, cif, aux, x, i;

    int main()
    {
        cin>>n;
        for (i=1;i<=n;i++)
        {
            cin>>x; aux=x;

            if (aux%2==0)
                a++;
            else
            {
                sum=0;
                while (aux>0)
                {
                    cif=aux%10;
                    sum=sum+cif;
                    aux=aux/10;
                }
                if (sum%13==0)
                    b++;
                else
                    c++;
            }
        }

        cout<<a<<' ';

        cout<<'\n'<<b<<' ';

        cout<<'\n'<<c<<' ';

        cin.close();
        cout.close();
        return 0;
}
```

PalindRoz – Andrici Cezar **100 puncte**

Marele Porc Rozaliu este foarte leneş. Este atât de leneş încat s-a gândit să angajeze un echipaj care să facă mizerie pentru el (este bine cunoscut faptul că porcilor le place mizeria). Deoarece echipajul care face mizerie cere prea mulţi bani, el încheie cu aceştia o înţelegere. Fiind foarte bun la informatică, Porcul Rozaliu a propus ca în schimbul rezolvării unei probleme, să primească o reducere. Spre surpriza lui, a primit următoarea problemă: pentru doua numere furnizate de echipaj, Marele Porc Rozaliu trebuie să se creeze cel mai mare palindrom folosind cifrele celor doua numere. Neştiind să facă problema, vă cere ajutorul pentru a nu se face de râs.

Cerinţă

Să se găsească cel mai mare palindrom format folosind cifrele a două numere date.

Date de intrare

Fişierul `palindroz.in` conţine două numere separate printr-un spaţiu, reprezentând valorile furnizate Marelui Porc Rozaliu de către echipajul care face mizerie.

Date de ieşire

Fişierul `palindroz.out` are o singură linie pe care se află numărul pe care trebuie să îl determine Marele Porc Rozaliu.

Restricţii şi precizări

- Cele două numere sunt din intervalul $[1, 10^8]$.
- Este posibil să nu trebuiască utilizate toate cifrele celor două numere.

Exemple:

palindroz.in	**palindroz.out**
1234 4321	43211234
78353 5877	875373578
989898 2220	9829289

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 4Mb / 2 Mb pentru stivă

Soluţie – Andrici Cezar

Se ține vectorul de frecvență **contor** de la 0 la 9 care va număra de cate ori apare fiecare cifra în cele două numere. După care vom lua fiecare cifră în ordine descrescatoare și o vom afișa de **contor[cifra]/2**. După vom afișa cea mai mare cifră care are un numar impar de apariții, apoi vom afișa în ordine crescătoare de data aceasta fiecare cifră de **contor[cifra]/2**.

```
#include <cstdio>

int x, y, contor[10], i, j, ok;

int main() {
	freopen("palindroz.in", "r", stdin);
	freopen("-palindroz.ok", "w", stdout);

		// citim cele doua numere
		scanf("%d %d", &x, &y);

		// tratam cazul cu 0
		if (x == 0 && y == 0) {
			printf("0\n");
			return 0;
		}

		// spargem x
		while (x > 0) {
			++contor[x%10];
			x /= 10;
		}

		// spargem y
		while (y > 0) {
			++contor[y%10];
			y /= 10;
		}

		// afisam prima jumatate din numar
		ok = false;
		for (i = 9; i >= 1; --i) {
			for (j = 1; j <= contor[i]/2; ++j) {
				ok = true;
				printf("%d", i);
			}
		}
		if (ok) {
```

```
            for (j = 1; j <= contor[0]/2; ++j) {
                printf("0");
            }
        }

        // afisam cifra maxima care apare de un nr impar de ori in centru
        for (i = 9; i >= 0; --i) {
            if (contor[i] % 2 == 1) {
                printf("%d", i);
                break;
            }
        }

        // afisam cealalta jumatate (de data asta in ordine inversa)
        if (ok) {
            for (j = 1; j <= contor[0]/2; ++j) {
                printf("0");
            }
        }

        for (i = 1; i <= 9; ++i) {
            for (j = 1; j <= contor[i]/2; ++j) {
                printf("%d", i);
            }
        }



    return 0;
}
```

Clasa a VI-a

Vestigii - Rosca Valentin **100 puncte**

Arheologul Dorel a ajuns în faţa uşii unui templu foarte vechi. El a observat că pe uşă erau gravate `n` numere pe un rând, iar pe următorul rând unul dintre simbolurile „+” sau „*”. Pentru a obţine codul de acces cu care poate intra în templu, Dorel trebuie ca, pornind de la şirul de `n` numere, să construiască alte `n-1` şiruri, în ordinea descrescătoare a lungimilor, de la şirul de lungime `n-1` la şirul de lungime `1`.
Şirul cu numărul de ordine `i`: a_1, a_2,…, a_{n-i+1} va fi folosit pentru a obţine şirul cu numărul de ordine `i+1` astfel: a_1 **op** a_2, a_2 **op** a_3, ..., a_{n-i} **op** a_{n-i+1}, unde **op** poate fi unul dintre simbolurile „+” sau „*”, acestea păstrându-şi semnificaţia matematică.
Pentru fiecare rezultat intermediar `a` **op** `b`, valoarea termenului din şir va fi:

- restul împărţirii lui la `45007`, dacă `a` **op** `b` este mai mare sau egal cu `45007`;
- `1`, dacă `a` **op** `b` este `0`;
- `a` **op** `b`, în rest;

Codul de acces este egal cu numărul de numere prime din toate cele `n` şiruri.

Cerinţa
Cunoscând `n` lungimea şirului iniţial, cele `n` numere ale sale precum şi codul `op`, corespunzător unuia dintre simbolurile „+” sau „*”, determinaţi codul de acces pentru intrarea în templu.

Date de intrare
Fişierul `vestigiu.in` conţine pe prima linie numărul natural nenul `n`, reprezentând lungimea şirului iniţial, pe cea de a doua linie `n` numere naturale nenule separate prin câte un spaţiu, iar pe a treia linie se află un număr reprezentând codul operaţiei `op` (`0` pentru „+”, `1` pentru „*”).

Date de ieşire
Fişierului `vestigiu.out` va conţine un număr natural reprezentând codul de acces.

Restricţii

- $2 \le N \le 3\ 000$;

- numerele şirului iniţial nu depăşesc 45007;
- 1 nu este considerat număr prim;

vestigii.in	vestigii.out	Explicaţii	
3 3 5 12 0	3	Şirul 1: **3 5** 12 Şirul 2: 8 **17** Şirul 3: 25 Codul de acces este 3 pentru că sunt trei numere prime 3, 5, 17.	
3 2 9 45006 1	1	Şirul 1: **2** 9 45006 Şirul 2: 18 44998 Şirul 3: 44845	(9*45006=405054, 405054%45007=44998) (18*44998=809964, 809964%45007=44845) Există un singur număr prim în cele 3 şiruri (2).

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 8 Mb pentru stivă

Soluţie – Roșca Valentin

```
#include <fstream>

using namespace std;

long ciur[45408],n,i,j,op,v[3001],prim;

int main()
{
    //fisierele de intrare/iesire
    ifstream fin("vestigii9.in");
    ofstream fout("vestigii9.ok");
    long nr=0; //contorul de numere prime

    //Ciurul lui Eratostene
    ciur[1]=1;
    for(i=2;i<=45007;i++) {
        if(ciur[i]==0)
            //marcam multiplii numarului prim
            for(j=i+i;j<=45007;j+=i)
                ciur[j]=1;
    }

    //citirea lui n, a sirului si a operatiei
    fin>>n;
    for(i=1;i<=n;i++)
        fin>>v[i];
    fin>>op;

    if(op==0) { //cazul in care operatie este "+"
        while(n>1) {
            v[n+1]=0;

            //mergem prin fiecare numar din sirul curent si vedem daca este
prim sau
            //nu si apoi il adunam cu termenul urmator pentru a afla urmatorul
sir
            for(i=1;i<=n;i++) {
                //adaugam numarul daca este prim
                if(ciur[v[i]]==0)
                    nr++;

                //formam noul numar din urmatorul sir
                v[i]=(v[i]+v[i+1]) % 45007;
                if(v[i]==0)
                    v[i]=1;
```

```
            }
            //trecem la urmatoarul sir
            n--;
        }
    }
    else { //cazul in care operatie este "*"
        while(n>1) {
            v[n+1]=0;

            //mergem prin fiecare numar din sirul curent si vedem daca este
prim sau
            //nu si apoi il inmultim cu termenul urmator pentru a afla
urmatorul sir
            for(i=1;i<=n;i++) {
                //adaugam numarul daca este prim
                if(ciur[v[i]]==0)
                    nr++;

                //formam noul numar din urmatorul sir
                v[i]=(v[i]*v[i+1]) % 45007;
                if(v[i]==0)
                    v[i]=1;
            }
            //trecem la urmatoarul sir
            n--;
        }
    }

    //verificam daca ultimul din ultimul sir este prim
    if(ciur[v[1]]==0)
        nr++;

    //afisarea
    fout<<nr;

    return 0;
}
```

Bcaptiv - Netedu Andrei **100 puncte**

𝔅-Darius, un mare olimpic la informatică, a fost răpit de extratereştrii şi este ţinut ostatic într-o închisoare. Închisoarea are forma unui caroiaj de dimensiune `n x m`, fiecare celulă reprezentând o cameră codificată printr-un număr natural distinct faţă de celelalte camere. 𝔅-Darius este foarte deştept şi realizează că, pentru a evada, trebuie să se deplaseze prin camerele închisorii astfel: din orice cameră poate merge doar într-o cameră vecină diferită de cea din care a venit, al cărei cod are număr maxim de divizori. În cazul în care există mai multe camere cu această proprietate o va alege pe cea cu codul maxim.
Două camere sunt vecine dacă au o latură comună. În momentul în care 𝔅-Darius ajunge a doua oară într-o cameră pe care a mai vizitat-o, poate accesa un portal care-l va teleporta înapoi pe Terra, dacă rosteşte parola. Parola reprezintă numărul de camere diferite parcurse de la prima vizitare a camerei în care se află până în momentul în care a ajuns a doua oară acolo.

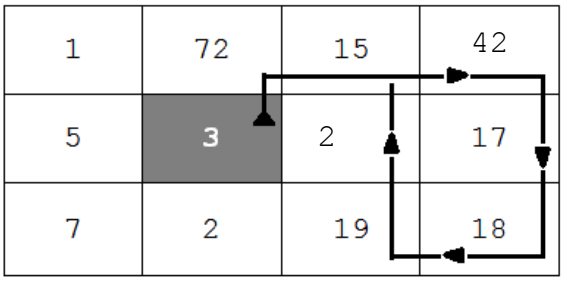
Cerinţa
Dându-se dimensiunile închisorii, valorile asociate camerelor şi poziţia iniţială a lui 𝔅-Darius, să se determine parola.

Date de intrare
Pe prima linie a fişierului de intrare `bcaptiv.in` se găsesc patru numere `n m L C`, reprezentând dimensiunile închisorii, linia `L` şi coloana `C` unde se află iniţial 𝔅-Darius. Pe fiecare dintre următoarele `n` linii se află `m` valori separate prin câte un spaţiu, reprezentând codurile camerelor.

Date de ieşire
Fişierul de ieşire `bcaptiv.out` va conţine parola care îl va salva pe 𝔅-Darius.

Restricţii şi precizări

- 2 ≤ n, m ≤ 100;
- Numerele atribuite camerelor sunt numere naturale nenule mai mici decât 250.000;
- Din închisoare nu se poate ieşi decât prin portal;
- Din nicio cameră ℬ-Darius nu se poate întoarce imediat în camera din care a venit.

bcaptiv.in	bcaptiv.out	Explicaţie
3 4 2 2 1 72 15 42 5 3 23 17 7 2 19 18	6	

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 8 Mb pentru stivă

Soluție – Filimon Marta

Vom reține 3 matrice cu urmatoarele semnificații:

- a[i][j]=codul camerei de pe linia i și coloana j
- nr_div[i][j]=numărul de divizori ai codului camerei de pe linia i și coloana j
- mom[i][j]=momentul de timp la care a ajuns prima dată în camera de pe linia i și coloana j

Cand vom ajunge într-o cameră de pe linia L și coloana C a închisorii, facând numărul de pași, pas, vom afla camera vecină a carui cod are un număr maxim de divizori (în caz că există mai multe camere cu această proprietate o vom alege pe cea care are codul maxim) și nu are momentul de timp la care ℬ-Darius a ajuns pentru prima dată la ea egal cu pas-1. În cazul în care camera aleasă a mai fost vizitată o data vom afișa valoarea cerută.

```
#include<fstream>
#define NMAX 110
#define VALMAX 250010

using namespace std;

ifstream f("bcaptiv.in");
ofstream g("bcaptiv.out");

int ciur[VALMAX], prime[VALMAX];
int a[NMAX][NMAX], nr_div[NMAX][NMAX], mom[NMAX][NMAX], n, m, L, C;
int di[4]={0, 0, 1, -1};
int dj[4]={-1, 1, 0, 0};

void Preprop()
{
    int i, j;

    ciur[1]=1;

    for (i=2; i<VALMAX; ++i)
    {
        if (!ciur[i])
        {
            prime[++prime[0]]=i;
```

```
            for (j=i+i; j<VALMAX; j+=i)
                ciur[j]=1;
        }
    }
}

inline int numara(int v)
{
    int ind=1, nr=1, aux=v, p;

    while (prime[ind]*prime[ind]<=v)
    {
        p=1;
        while (aux%prime[ind]==0)
        {
            aux/=prime[ind];
            ++p;
        }
        nr*=p;++ind;
    }

    if (aux!=1) nr*=2;

    return nr;
}

void Citeste()
{
    int i, j;

    f>>n>>m>>L>>C;

    for (i=1; i<=n; ++i)
        for (j=1; j<=m; ++j)
        {
            f>>a[i][j];
            nr_div[i][j]=numara(a[i][j]);
        }
}

void Solve()
{
    int i, j, k, gata=0, pas=1, im, jm, mx, nrmax;

    for (i=0; i<=n+1; ++i)
        for (j=0; j<=m+1; ++j)
            mom[i][j]=-1;

    mom[L][C]=1;
    while (!gata)
    {
        mx=-1; nrmax=-1;
        for (k=0; k<4; ++k)
        {
            i=L+di[k];
            j=C+dj[k];
            if (mom[i][j]!=pas-1)
            {
                if (nr_div[i][j]>mx)
                {
                    mx=nr_div[i][j]; nrmax=a[i][j];
```

```
                    im=i; jm=j;
                }
                else
                    if (nr_div[i][j]==mx && a[i][j]>nrmax)
                    {
                        mx=nr_div[i][j]; nrmax=a[i][j];
                        im=i; jm=j;
                    }
            }
        }
        L=im; C=jm;
        ++pas;
        if (mom[im][jm]!=-1)
            gata=pas-mom[im][jm];
        mom[im][jm]=pas;
    }
    g<<gata<<"\n";
}

int main()
{
    Preprop();

    Citeste();

    Solve();

    f.close();
    g.close();
    return 0;
}
```

Clasa a VII-a

Party – Filimon Marta **100 puncte**

Ruxandra a hotărât să îşi sărbatorească ziua de naştere împreună cu prietenii săi. Cum ea este o fată populară, la petrecere vor veni toţi prietenii ei de pe Facebook. Aceasta doreşte ca fiecare invitat să îşi asculte cântecul favorit. Ruxandra îşi numerotează cei `1.000.000` de prieteni de la `1` la `1.000.000` şi îşi formeaza o listă cu toate cântecele posibile, atribuind fiecăruia câte un număr. Pentru fiecare prieten `n`, melodia lui favorită este `p`, unde `p` se calculează conform formulei:

$$\underbrace{4444...4}_{\text{de 2n ori}} - \underbrace{8888...8}_{\text{de n ori}} = p^2$$

Cerinţă

În ziua petrecerii, cea mai bună prietenă a Ruxandrei, aflată în listă pe poziţia n, vrea să asculte cântecul său preferat. Deoarece ea este ocupată cu pregatirile vă roagă pe voi să-l identificaţi.

Date de intrare

În fişierul de intrare `party.in` pe prima linie se află un singur număr natural `n`, cu semnificaţia din enunţ.

Date de ieşire

În fişierul de ieşire `party.out` se va găsi un singur număr natural `p`, cu semnificaţia din enunţ.

Restricţii

- $1 \le n \le 1.000.000$
- Pentru `20%` din teste $n \le 4$
- Pentru alte `20%` din teste $n \le 9$

Exemplu

party.in	party.out	Expicaţii
2	66	$4444-88= 4.356=66^2$

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:**4Mb / 2 Mb pentru stivă

Soluție – Filimon Marta

Pentru a ajunge la rezultatul dorit vom efectua următoarele calcule:

$$\underbrace{444\ldots}_{2n\text{ ori}}4-\underbrace{888\ldots}_{n\text{ ori}}8=p^2$$

$$4*(\underbrace{111\ldots}_{2n\text{ ori}}1-2*\underbrace{111\ldots}_{n\text{ ori}}1)=p^2$$

$$4*[10^{2n-1}+10^{2n-2}+\cdots+1-2*(10^{n-1}+10^{n-2}+\cdots+1)]=p^2$$

$$4*\left(\frac{10^{2n}-1}{10-1}-2*\frac{10^n-1}{10-1}\right)=p^2$$

$$4*\left(\frac{10^{2n}-1-2*10^n+2}{9}\right)=p^2$$

$$4*\left(\frac{10^{2n}-2*10^n+1}{9}\right)=p^2$$

$$2^2*\frac{(10^n-1)^2}{3^2}=p^2$$

$$\left(2*\frac{10^n-1}{3}\right)^2=p^2$$

$$2*\frac{10^n-1}{3}=p$$

Se observa cu uşurinţă că $p=\underbrace{66\ldots6}_{n\text{ ori}}$,

Dpiv – Netedu Andrei **100 puncte**

Se dau două şiruri de numere naturale nenule `a` şi `b` cu `n`, respectiv `m` elemente. Din aceste două şiruri se creează un nou şir `c` astfel: se înmulteşte primul număr din `a`, pe rând, cu toate numerele din `b`; apoi se înmulteşte al doilea număr din `a`, pe rând, cu toate numerele din `b` etc. Se obţine astfel şirul `c`: $c_1=a_1*b_1$; $c_2=a_1*b_2$; …, $c_m=a_1*b_m$; $c_{m+1}=a_2*b_1$; $c_{m+2}=a_2*b_2$; ..., $c_{n*m}=a_n*b_m$.

Cerinţa

Dându-se un număr `x` care apare în şirul `c` şi o poziţie `p`, determinaţi o aranjare a şirurilor `a` şi `b` astfel încât prima apariţie a numărului `x` în şirul `c` să se afle pe poziţia `p`.

Date de intrare

Fişierului de intrare `dpiv.in` va conţine pe prima linie patru numere naturale `n m x p` separate prin câte un spaţiu, cu semnificaţia din enunţ. A doua linie va conţine elementele şirului `a`, iar a treia linie va conţine elementele şirului `b`, separate prin câte un spaţiu.

Date de ieşire

Fişierul de ieşire `dpiv.out` va conţine pe prima linie o posibilă aranjare a şirului `a`, iar pe a doua linie o posibilă aranjare a şirului `b`.

Restricţii şi precizări

- Se garantează că există întotdeauna soluţie;
- Elementele şirurilor sunt distincte, însă cele două şiruri pot avea elemente comune;
- $1\le n, m \le 10000, 1\le p, x\le 10^9$;
- Elementele şirurilor nu vor depăşi 10^8.

dpiv.in	dpiv.out	Explicaţie

4 5 10 3
2 7 5 1
9 10 5 2 1
	2 7 5 1
	9 2 5 10 1
		Şirul creat din soluţia posibilă este
		2*9, 2*2, **2*5=10**, 2*10, 2*1, 7*9 ...; se observă că prima apariţie a numărului 10 este pe poziţia 3.
		Două alte soluţii posibile sunt:
		a=(5,2,7,1), b=(1,10,2,5,9);
		a=(1,5,2,7), b=(9,5,10,1,2);

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 8 Mb pentru stivă

Soluție – Filimon Marta

```
#include<fstream>
#include<algorithm>
#define NMAX 10010

using namespace std;

ifstream f("dpiv.in");
ofstream g("dpiv.out");

int a[NMAX], b[NMAX], A[NMAX], n, m, p, ia, ib, x;

void Citeste()
{
    int i;

    f>>n>>m>>x>>p;

    for (i=1; i<=n; ++i) f>>a[i];

    for (i=1; i<=m; ++i) f>>b[i];
}

int cauta(int x)
{
    int st=1, dr=m, mij;

    while (st<=dr)
    {
        mij=(st+dr)/2;

        if (b[mij]==x) return mij;
        else if (x<b[mij]) dr=mij-1;
        else st=mij+1;
    }

    return 0;
}

void Scrie()
{
    int i;

    for (i=1; i<=n; ++i) g<<A[i]<<" ";
    g<<"\n";

    for (i=1; i<=m; ++i) g<<b[i]<<" ";
    g<<"\n";

}

void Solve()
{
    int i, poz, pA=1, uA=n;

    sort(b+1, b+m+1);

    if (p%m==0)
    {
        ia=p/m;
        ib=m;
```

```
    }
    else
    {
        ia=p/m+1;
        ib=p%m;
    }

    for (i=1; i<=n; ++i)
        if (x%a[i]==0)
        {
            if (cauta(x/a[i]))
            {
                A[uA]=a[i];
                uA--;
            }
            else
            {
                A[pA]=a[i];
                pA++;
            }
        }
        else A[pA++]=a[i];


    swap(A[n], A[ia]);

    poz=cauta(x/A[ia]);

    swap(b[poz], b[ib]);

    Scrie();
}

int main()
{
    Citeste();

    Solve();

    f.close();
    g.close();
    return 0;
}
```

Clasa a IX-a

Pisici – Luca Roxana **100 puncte**

Trei pisici au ieşit la vânătoare de şoricei. Ele au vânat `n` şoricei, i-au etichetat cu câte un număr natural, în ordinea în care au fost prinşi. Ca să nu se certe între ele de la împărţirea şoriceilor, au hotărât să-i distribuie după următoarele reguli:

- Prima pisică va lua acei şoricei care au etichetele numere cu cel puţin două cifre iar ultimile două cifre ale acestora sunt pare
- A doua pisică va lua dintre şoriceii rămaşi pe cei pentru care suma cifrelor mai mari sau egale cu 5 ale numărului de pe etichetă este divizibilă cu `11`
- A treia pisică va lua toţi şoriceii rămaşi după ce primele două şi-au primit partea.

Cerinţă

Dându-se numărul şoriceilor şi etichetele lor, să se determine câţi şoricei va mânca fiecare pisică.

Date de intrare

În fişierul de intrare `pisici.in` se va afla pe prima linie numărul natural `n`, numărul de şoricei, iar pe următoarea linie `n` numere naturale reprezentând numerele de pe etichetele şoriceilor.

Date de ieşire

În fişierul `pisici.out` se va afişa pe prima linie `n1`, numărul de şoricei pe care îi va mânca prima pisică, pe a doua linie se va afişa `n2`, numărul de şoricei pe care îi va mânca a doua pisică, pe a treia linie se va afişa `n3`, numărul de şoricei pe care îi va mânca a treia pisică.

Restricţii şi precizări

- `1 ≤ n ≤ 1000`
- Numerele de pe etichetele şoriceilor vor fi cuprinse între `1` şi `10000`.
- **În cazul în care nu ştiţi să aflaţi numărul de şoricei mâncaţi de o pisică, vă rugăm să afişaţi `0` pe linia corespunzătoare ei.**
- Se vor acorda puctaje parţiale astfel: pentru primele două rezultate corecte câte `40%` din punctajul testului şi `20%` pentru ultimul rezultat corect.

pisici.in	pisici.out	Explicaţii
3 58 645 2	0 1 2	Fiecare pisică manâncă un singur şoricel: • Prima pisică nu va mânca niciun şoricel • A doua pisică mănâncă soricelul cu eticheta `645` deoarece este singurul număr rămas pentru care suma cifrelor mai mari ca `5` este divizibilă cu `11`(`6+5=11`, `11:11`) • Celei de-a treia pisici îi râman şoricelul cu eticheta `2` şi cel cu eticheta `58`.
6 77831 3 26 41 342 7	2 1 3	

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 4Mb / 2 Mb pentru stivă

Soluție – Luca Roxana

Pentru fiecare pisică vom avea asociată o variabilă care contorizează numărul de șoricei care îi revin.
Vom parcurge etichetele soriceilor. Pentru o eticheta `x`, aflată pe poziția `i` în fișierul de intrare, vom testa:

1) Dacă `x` are ultimile doua cifre pare numărul de șoricei care îi revin primei pisici este mărit cu unu.
2) Daca **suma cifrelor** lui `x` mai mari ca 4 este **divizibilă cu `11`** și nu îndeplinește condiția `1)` atunci numărul de șoricei care îi revin celei de a doua pisici este mărit cu unu.
3) Dacă nu îndeplinește nicio condiție de mai sus atunci numărul de șoricei care îi ultimii pisici este mărit cu unu.

```
#include <fstream>

using namespace std;

ifstream cin("pisici.in");
ofstream cout("pisici.out");

int a=0, b=0, c=0, n, sum, cif, aux, x, i;

int main()
{
    cin>>n;
    for (i=1;i<=n;i++)
    {
        cin>>x; aux=x;

        cif=aux/10;
        if (aux>9 && aux%2==0 && cif%2==0) ++a;
        else
        {
            sum=0;
            while (aux>0)
            {
                cif=aux%10;
                if (cif>4) sum=sum+cif;
```

```
                aux=aux/10;
            }
            if (sum!=0 && sum%11==0) b++;
            else c++;
        }
    }

    cout<<a<<"\n"<<b<<"\n"<<c<<"\n";

    cin.close();
    cout.close();
    return 0;
}
```

Bcaptiv – Netedu Andrei **100 puncte**

Ƀ-Darius, un mare olimpic la informatică, a fost răpit de extratereştrii şi este ţinut ostatic într-o închisoare. Închisoarea are forma unui caroiaj de dimensiune `n x m`, fiecare celulă reprezentând o cameră codificată printr-un număr natural distinct faţă de celelalte camere. Ƀ-Darius este foarte deştept şi realizează că, pentru a evada, trebuie să se deplaseze prin camerele închisorii astfel: din orice cameră poate merge doar într-o cameră vecină diferită de cea din care a venit, al cărei **cod oglindit** are număr maxim de divizori. În cazul în care există mai multe camere cu această proprietate o va alege pe cea cu **codul oglindit** maxim. De exemplu, camera care are codul `456` va avea **codul oglindit** `654`.

Două camere sunt vecine dacă au o latură comună. În momentul în care Ƀ-Darius ajunge a doua oară într-o cameră pe care a mai vizitat-o, poate accesa un portal care-l va teleporta înapoi pe Terra, dacă rosteşte parola. Parola reprezintă numărul de camere diferite parcurse de la prima vizitare a camerei în care se află până în momentul în care a ajuns a doua oară acolo.

Cerinţa

Dându-se dimensiunile închisorii, valorile asociate camerelor şi poziţia iniţială a lui Ƀ-Darius, să se determine parola.

Date de intrare

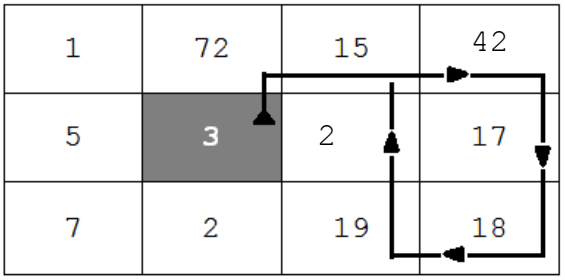
Pe prima linie a fişierului de intrare `bcaptiv.in` se găsesc patru numere `n m L C`, reprezentând dimensiunile închisorii, linia `L` şi coloana `C` unde se află iniţial Ƀ-Darius. Pe fiecare dintre următoarele `n` linii se află `m` valori separate prin câte un spaţiu, reprezentând codurile camerelor.

Date de ieşire

Fişierul de ieşire `bcaptiv.out` va conţine parola care îl va salva pe Ƀ-Darius.

Restricţii şi precizări

- $2 \le n, m \le 100$;
- Numerele atribuite camerelor sunt numere naturale nenule mai mici decât `250.000`;
- Din închisoare nu se poate ieşi decât prin portal;
- Din nicio cameră Ƀ-Darius nu se poate întoarce imediat în camera din care a venit.

bcaptiv.in	**bcaptiv.out**	**Explicaţie**
3 4 2 2 1 27 51 24 5 3 32 71 7 2 91 81	6	În tabelul de mai jos am folosit codul oglindit al camerelor. 

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 8 Mb pentru stivă

Soluție – Filimon Marta

Vom reține 3 matrice cu urmatoarele semnificații:

- a[i][j]=oglinditul codului camerei de pe linia i și coloana j
- nr_div[i][j]=numărul de divizori ai codului camerei de pe linia i și coloana j
- mom[i][j]=momentul de timp la care a ajuns prima dată în camera de pe linia i și coloana j

Cand vom ajunge într-o cameră de pe linia L și coloana C a închisorii, facând numărul de pași, pas, vom afla camera vecină a carui cod are un număr maxim de divizori (în caz că există mai multe camere cu această proprietate o vom alege pe cea care are codul maxim) și nu are momentul de timp la care ℬ-Darius a ajuns pentru prima dată la ea egal cu pas-1. În cazul în care camera aleasă a mai fost vizitată o data vom afișa valoarea cerută.

```
#include<fstream>
#define NMAX 110
#define VALMAX 250010

using namespace std;

ifstream f("bcaptiv.in");
ofstream g("bcaptiv.out");

int ciur[VALMAX], prime[VALMAX];
int a[NMAX][NMAX], nr_div[NMAX][NMAX], mom[NMAX][NMAX], n, m, L, C;
int di[4]={0, 0, 1, -1};
int dj[4]={-1, 1, 0, 0};

void Preprop()
{
    int i, j;

    ciur[1]=1;

    for (i=2; i<VALMAX; ++i)
    {
        if (!ciur[i])
        {
            prime[++prime[0]]=i;
            for (j=i+i; j<VALMAX; j+=i)
                ciur[j]=1;
        }
    }
}

inline int numara(int v)
{
    int ind=1, nr=1, aux=v, p;

    while (prime[ind]*prime[ind]<=v)
    {
        p=1;
        while (aux%prime[ind]==0)
        {
            aux/=prime[ind];
            ++p;
        }
        nr*=p;++ind;
    }

    if (aux!=1) nr*=2;
```

```
    return nr;
}

void Citeste()
{
    int i, j, x;

    f>>n>>m>>L>>C;

    for (i=1; i<=n; ++i)
        for (j=1; j<=m; ++j)
        {
            f>>x;
            while (x)
            {
                a[i][j]=a[i][j]*10+x%10;
                x=x/10;
            }
            nr_div[i][j]=numara(a[i][j]);
        }
}

void Solve()
{
    int i, j, k, gata=0, pas=1, im, jm, mx, nrmax;

    for (i=0; i<=n+1; ++i)
        for (j=0; j<=m+1; ++j)
            mom[i][j]=-1;

    mom[L][C]=1;
    while (!gata)
    {
        mx=-1; nrmax=-1;
        for (k=0; k<4; ++k)
        {
            i=L+di[k];
            j=C+dj[k];
            if (mom[i][j]!=pas-1)
            {
                if (nr_div[i][j]>mx)
                {
                    mx=nr_div[i][j]; nrmax=a[i][j];
                    im=i; jm=j;
                }
                else
                    if (nr_div[i][j]==mx && a[i][j]>nrmax)
                    {
                        mx=nr_div[i][j]; nrmax=a[i][j];
                        im=i; jm=j;
                    }
            }
        }
        L=im; C=jm;
        ++pas;
        if (mom[im][jm]!=-1)
            gata=pas-mom[im][jm];
        mom[im][jm]=pas;
    }
    g<<gata<<"\n";
}
```

```
int main()
{
    Preprop();

    Citeste();

    Solve();

    f.close();
    g.close();
    return 0;
}
```

Problemele de concurs 2012

Clasa a V-a

Automat-Filimon Marta şi Valentin Roşca **100 puncte**

Marius a primit un premiu în bani deoarece a obţinut un rezultat foarte bun la concursul Moisil++. Premiul este format din bancnote de `patru` lei, de `trei` lei, de `doi` lei şi de `un` leu.
Lui Marius îi place foarte mult ciocolata aşa că s-a dus la automatul şcolii să cumpere ciocolăţele pentru a le oferi câtor mai mulţi colegi. El ştie că o ciocolată costă 4 lei şi că automatul acceptă doar suma exactă pentru a o cumpăra.

Cerinţă

Ajutaţi-l pe Marius să afle numărul maxim de ciocolăţele pe care le poate cumpăra ştiind câte bancnote de fiecare tip are la dispoziţie.

Date de intrare

Din fişierul text `automat.in` se citesc patru numere `x y z w`, separate prin câte un spaţiu, numere ce reprezintă câte monezi de `patru` lei, de `trei` lei, de `doi` lei şi de `un` leu, primite ca premiu .

Date de ieşire

In fişierul `automat.out` se va afla un singur număr `C` care este egal cu numărul maxim de ciocolăţele pe care le poate cumpăra Marius de la automat.

Restricţii şi precizări

$0 \le x, y, z, t \le 2000$

automat.in	automat.out	Explicaţii
3 3 5 2	7	Vom putea cumpăra 3 ciocolăţele folosind doar bacnote de patru lei. Vom putea cumpăra 2 ciocolăţele folosind bacnote de trei, respectiv un leu. Vom putea cumpăra 2 ciocolăţele folosind doar bacnote de doi lei.

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 2Mb / 1 Mb pentru stivă

Soluţie-Filimon Marta

```
#include<fstream>

using namespace std;

ifstream f("automat.in");
ofstream g("automat.out");

int x, y, z, w, numar;

int main()
{
      f>>x>>y>>z>>w;

      /* cu bacnotele de 4 lei pot cumpara deja x ciocolate, unde x reprezinta numarul
de bacnote de valoare 4 lei*/

      numar=x;

      /*valoare de 4 lei= 3 lei+1 leu =>  la numarul de ciocolate cumparate pot adauga
minimul dintre numarul de bacnote de 1 si numarul de bacnote de 3*/

      if (y<w)
      {
            numar=numar+y;
            w=w-y; y=0;
      }
      else
      {
```

```
            numar=numar+w;
            y=y-w; w=0;
      }

      /*valoare de 4 lei= 2 lei+2 lei =>  la numarul de ciocolate cumparate  pot adauga
jumatate din numarul de bacnote de 2 lei*/

      numar=numar+z/2;

      /*daca avem un numar impar de bacnote de 2 lei atunci ne va ramane o bacnota de 2
lei cu care mai pot cumpara o ciocolata in cazut in care mai avem ramase cel putin 2
bacnote de un leu*/

      if (z%2==1 && w>1)
      {
            ++numar;
            w=w-2;
      }

      /*valoare de 4 lei= 1 leu+1 leu+1 leu+1 leu=> la numarul de ciocolate cumparate
pot adauga un sfert din numarul de bacnote de un leu ramase*/

      numar=numar+w/4;

      g<<numar<<"\n";

      f.close();
      g.close();
      return 0;
}
```

Şir-Roşca Valentin Bogdan

100 puncte

Se dă şirul: `1,2,3,7,8,9,13,14,15,19,…,` aranjat sub forma unei jumătăţi de piramidă (imaginea din dreapta) astfel încât pe linia `n` se află următoarele `n` numere din şir. Numerotarea liniilor începe de la `1` iar coloanele sunt numerotate de asemenea începând cu `1`, de la stânga la dreapta (de exemplu, valoarea `14` este pe linia `4`, coloana `2`)

```
 1
 2  3
 7  8  9
13 14 15 19
.............
```

Cerinţă

Dându-se un număr natural `x` să se verifice dacă `x` se află în şir şi, dacă da, să se determine poziţia acestuia în cadrul şirului precum şi linia şi coloana pe care se află acesta în piramida construită conform enunţului.

Date de intrare

Din fişierul text `sir.in` se citeşte de pe prima linie numărul natural `x`.

Date de iesire

Rezultatele se vor scrie în fişierul text `sir.out` astfel:

- pe prima linie se va scrie `1` dacă numărul se află în şir, respectiv `0` dacă numărul nu face parte din şir
- pe a doua linie se va scrie, dacă există, un număr `P` care reprezintă pozitia lui `x` în şirul dat
- pe a treia linie se vor scrie două numere `L C`, unde `L` este linia iar `C` coloana pe care se află elementul `x` în piramida construită conform enunţului

Restricţii şi precizări

- `x < 2.000.000.000`
- pentru prima cerinţă se acordă `20%` din punctaj
- pentru a doua cerinţă se acordă `40%` din punctaj
- pentru a treia cerinţă se acordă `40%` din punctaj

sir.in	sir.out
12	0
14	1 8 4 2

Timp maxim de execuţie/test: 0,1 secundă **Memorie disponibilă:** 2Mb / 1Mb pentru stivă

Soluţie- Roşca Valentin Bogdan

In primul rînd se observa faptul ca şirul este format după regula:
6*0+1, 6*0+2, 6*0+3, 6*1+1, 6*1+2, 6*1+3, 6*2+1, 6*2+2, 6*2+3, 6*3+1, …, 6*k+1, 6*k+2, 6*k+3, ...

Prima cerinţă:
Pentru a testa dacă numărul x se afla în şir se face restul împărţirii lui la 6. În cazul în care restul este 1, 2 sau 3, atunci el se afla în şir.

A doua cerinţă:
Pentru a afla al câtelea termen este în şir vom genera şirul cu un while pînă cînd vom ajunge la termenul x.
O alta modalitate de a afla poziţia lui în şir este sa împărţim x la 6 si sa luam partea intreaga. Aşa aflăm cate grupe de 6*k+1, 6*k+2, 6*k+3 sunt inaintea lui x. Pentru a afla pozitia lui vom înmulţi numarul de grupe cu 3 (3 termeni pe grupa) şi adunam totul cu poziţia lui x in grupa în care se află, adica restul lui x la împarţirea cu 6. Deci formula finală va fi egală cu (x/6)*3 + x%6.

A treia cerinţa:
Pentru a-i afla poziţia lui în piramida vom afla mai întîi linia pe care se află x. Ştim de la punctul al doilea poziţia lui în şir, aşa că vom afla cate linii complete sunt înaintea lui, ţinînd cont de următoarea inegalitate dubla:
n*(n+1)/2 < x <= (n+1)(n+2)/2. unde n reprezintă numărul de linii complete care se afla înaintea lui x si care poate fi aflat printr-un while. Deci linia pe care se afla este egală cu n+1. Pentru a afla coloana pe care se află, vom scădea din poziţia lui x în şir numărul de termeni de pe liniile anterioare, astfel rămânând doar numărul de termeni de pe linia curentă până la x. În concluzie coloana pe care se află x este egală cu b - n*(n+1)/2 unde b este rezultatul obţinut la punctul al doilea.

Program:

```
#include<stdio.h>
long x, s, ad, poz, ok,r;
int main()
{
	freopen("sir.in","r",stdin);
	freopen("sir.out","w",stdout);
	scanf("%ld",&x);
	r=x%6;
	if(1<=r && r<=3)
		ok=1;
	printf("%ld\n",ok);
	if(ok)
	{
		poz=(x/6)*3+x%6;
		printf("%ld\n",poz);
		s=1;ad=2;
		while (s<=poz)
		{
			s=s+ad;
			ad++;
		}
		s=s-ad+1;
		if(s==poz)
		{
			ad--;
			s=1;
		}
		printf("%ld %ld",ad-1,poz-s);
	}
	return 0;
}
```

Clasa a VI-a

Armonie –Filimon Marta Diana **100 de puncte**

Ruxandra şi-a cumpărat o casă magică în Londra care poate influenţa pozitiv sau negativ delegaţia României la Jocurile Olimpice. Casa are `n` etaje şi pe fiecare etaj sunt `m` camere în care vor sta prietenii Ruxandrei. Pentru fiecare camera sunt cunoscute numărul de obiecte norocoase şi numărul de obiecte ghinioniste. Ca să nu influenţeze sportivii români, casa Ruxandrei trebuie sa fie armonioasă.
O casă este armonioasă daca fiecare camera este în armonie. O cameră este în armonie în cazul în care este locuită şi dacă suma dintre numărul de obiecte ghinioniste din acea cameră şi numarul de camere vecine cu aceasta, care au un număr de obiecte norocoase nenul este `k`, sau dacă este nelocuită. Vecinii unei camere sunt camerele aflate la stânga, dreapta, sus şi jos faţă de aceasta.

Cerinţă
Să se determine dacă casa Ruxandrei este armonioasă, iar în caz contrar să se determine configuraţia armonioasă a casei aruncând doar obiectele ghinioniste sau cumpărând obiecte ghinioniste.

Date de intrare
Pe prima linie a fişierului armonie.in se vor afla `n`, `m` şi `k` cu semnificaţiile din enunţ. Apoi pe următoarele `n` linii se vor afla `2*m` numere cu semnificatia. Al `2*j-1` lea (`1≤j≤m`)număr reprezintă numărul de obiecte norocoase şi al `2*j` lea număr reprezintă numărul de obiecte ghinioniste din camera `j`, reprezentând configuraţia iniţială a casei Ruxandrei.

Date de ieşire
Pe prima linie a fişierului armonie.out se va afla `DA`, în cazul în care casa Ruxandrei este în armonie, iar `NU` in caz contrar. În cazul în care casa Ruxandrei nu este armonioasă se se afişeze configuraţia armonioasă a casei.

Restricţii şi precizări
`1≤n, m≤500, 5≤k≤100`
Dacă numărul de obiecte norocoase dintr-o cameră este egal cu `0` atunci şi numărul de obiecte ghinioniste va fi egal cu `0`.
Dacă o cameră are `0` obiecte norocoase atunci ea este armonioasă.

armonie.in	armonie.out	Explicarie
4 5 7 0 0 3 6 0 0 0 0 1 6 2 6 1 3 5 4 1 5 2 5 0 0 4 5 1 4 0 0 0 0 6 7 0 0 1 6 0 0 3 7	DA	Casa este armonioasă.
4 5 7 0 0 3 5 0 0 0 0 **1 9** 2 6 **1 1** 5 4 1 5 2 5 0 0 4 5 **1 5** 0 0 0 0 6 7 0 0 **1 9** 0 0 **3 13**	NU 0 0 3 6 0 0 0 0 1 6 2 6 1 3 5 4 1 5 2 5 0 0 4 5 1 4 0 0 0 0 6 7 0 0 1 6 0 0 3 7	Am evidenţiat în fişierul armonie.in camerele în care vom modifica numărul de obiecte ghinioniste.

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 2 Mb pentru stivă

Soluţie–Filimon Marta Diana

Vom reţine numărul de obiecte norocoase din a j-a cameră de la etajul i în x[i][j], iar analog vom retine obiectele ghinioniste din a j-a cameră de la etajul i în y[i][j].
Vom parcurge matricea x, iar atunci când găsim x[i][j]≠0 vom verifica dacă camera este în armonie, în cazul în care camera nu este în armonie atunci vom modifica y[i][j]. O cameră este în armonie dacă y[i][j]+numărul de elemente nenule al mulţimii {x[i][j-1], x[i][j+1], x[i+1][j], x[i-1][j]}=k. Dacă această sumă (să o notăm cu sum) este diferită de k. Atunci distingem 2 cazuri:

1. sum<k atunci avem nevoie de mai multe obiecte ghinioniste => y[i][j]=y[i][j]+k-sum;
2. sum>k atunci avem trebuie să aruncăm obiecte ghinioniste => y[i][j]=y[i][j]-(sum-k);

```
#include<fstream>
#define NMAX 510

using namespace std;

int x[NMAX][NMAX], y[NMAX][NMAX], n, m, k, i, j, nr=0, sum, L;
```

```
int di[4]={0, 0, -1, 1};
int dj[4]={-1, 1, 0, 0};

ifstream f("armonie.in");
ofstream g("armonie.out");

int main()
{
      f>>n>>m>>k;
      for (i=1; i<=n; ++i)
            for (j=1; j<=m; ++j) f>>x[i][j]>>y[i][j];
      for (i=1; i<=n; ++i)
            for (j=1; j<=m; ++j)
            if (x[i][j])
            {
                  sum=y[i][j];
                  for (L=0; L<4; ++L)
                        if (x[i+di[L]][j+dj[L]]) ++sum;
                  if (sum!=k)
                  {
                        if (sum>k) y[i][j]-=sum-k;
                              else y[i][j]+=k-sum;
                        ++nr;
                  }
            }

      if (!nr) g<<"DA\n";
      else
      {
            g<<"NU\n";
            for (i=1; i<=n; ++i)
            {
                  g<<x[i][1]<<" "<<y[i][1];
                  for (j=2; j<=m; ++j) g<<" "<<x[i][j]<<" "<<y[i][j];
                  g<<"\n";
            }
      }
      f.close();
      g.close();
      return 0;
}
```

Clasa a VI-a

JO-Cohal Alexandru **100 puncte**

Alf şi Zilla plănuiesc să se uite anul acesta la televizor la Jocurile Olimpice (J.O.) şi să urmărească cu interes toate probele sportive. Cei doi vor să ştie la finalul competiţiei câte medalii a obţinut fiecare ţară participantă. Alf şi Zilla se hotărăsc să calculeze singuri acest clasament, deoarece nu vor să se bazeze pe televizorul lor, care e de pe vremea bunicului Torbjørn şi care se poate strica exact în momentul afişării acestui clasament pe ecran.
Pentru a-şi uşura munca, Alf şi Zilla codifică cu câte un număr `s` fiecare probă sportivă din cadrul J.O.. Fiecare ţară participantă la o probă sportivă este reprezentată numai de un singur sportiv. Astfel, Alf şi Zilla codifică fiecare sportiv cu câte un număr `t`, semnificând ţara pe care o reprezintă. Cei doi nu uită aceste numere asociate, şi le folosesc pe aceleaşi pe tot parcursul competiţiei.
Pe toată durata J.O., pentru fiecare evoluţie a unui concurent, Alf şi Zilla notează într-un fişier câte o secvenţă de forma `s t pi pz`, unde `s` reprezintă codul probei sportive, `t` reprezintă codul ţării participantului, iar `pi` şi `pz` formează punctajul obţinut, `pi` fiind partea întreagă iar `pz` partea zecimală. Este cunoscut faptul că un concurent poate evolua de mai multe ori la o probă, punctajul său final fiind cel maxim.
La sfârşitul unei probe, sportivul ce obţine punctaj maxim primeşte *medalia de aur*, sportivul ce obţine al doilea punctaj valoric primeşte medalia de *argint*, iar sportivul ce obţine al treilea punctaj valoric primeşte medalia de *bronz*. Pentru că un sportiv este identificat prin codul ţării de provenienţă, spunem că ţara este premiată cu una din medaliile de *aur*, *argint* sau *bronz*.

Cerinţă
Să se determine numărul de probe sportive distincte şi numărul de ţări participante din competiţie precum şi câte medalii de fiecare tip a obţinut fiecare ţară participantă.

Date de intrare
Fişierul de intrare `jo.in` conţine pe prima linie numărul natural `n` reprezentând numărul total de evoluţii care au avut loc la Jocurile Olimpice din acest an. Pe următoarele `n` linii se află câte patru numere naturale, separate prin câte un spaţiu, `s t pi pz` cu semnificaţia din enunţ.

Date de ieşire
Fişierul de ieşire `jo.out` va conţine pe prima linie două numere naturale `NrS` şi `NrT` separate printr-un spaţiu, reprezentând numărul de probe sportive distincte din competiţie, respectiv numărul de ţări distincte participante la Jocurile Olimpice. Următoarele `NrT` linii vor conţine câte patru numere naturale, separate prin câte un spaţiu, `t au ag br`, unde `au`, `ag` şi `br` reprezintă numărul de medalii de aur, argint respectiv bronz obţinute de ţara cu codul `t`, aceste linii fiind ordonate crescător în funcţie de codul ţării, `t`.

Restricţii şi precizări
$3 \le n \le 300$; $1 \le s, t \le 500$; $0 \le pi \le 200$; $0 \le pz \le 9$
Pentru fiecare probă sportivă vor fi cel puţin `3` participanţi.
Prin *parte întreagă* se înţelege numărul din faţa virgulei unui numar real, iar prin *parte zecimală* se înţelege numărul de după virgulă.
Pentru fiecare probă sportivă vor fi acordate exact trei medalii. Nu vor exista sportivi cu acelaşi punctaj pe primele trei poziţii.
Codurile probelor sportive respectiv codurile ţărilor particpante nu sunt neapărat numere consecutive.
Se acordă `30%` din puctaj pentru afişarea corectă a numărului de probe sportive distincte din competiţie şi a numărului de ţări distincte participante la Jocurile Olimpice.

jo.in	jo.out	Explicaţie
8 1 2 1 5 3 4 5 5 1 5 2 5 1 2 4 1 3 7 6 2 1 7 10 5 3 10 6 1 1 10 0 5	2 5 2 0 1 0 4 0 0 1 5 0 0 1 7 2 0 0 10 0 1 0	La competiţie au loc 2 probe sportive (având codurile 1 şi 3) şi participă 5 ţări (având codurile 2, 4, 5, 7 şi 10). • Ţara 2 are o singură medalie, de argint, la proba 1 (cu punctajul 4.1) (sportivul evoluează de două ori, obţinând punctajele 1.5 şi 4.1, punctajul final pentru el fiind 4.1). • Ţara 4 are o singură medalie, de bronz, la proba 3 (cu punctajul 5.5). • Ţara 5 are o singură medalie, de bronz, la proba 1 (cu punctajul 2.5). • Ţara 7 are două medalii de aur, una la proba 1 (cu punctajul 10.5), iar cealaltă la proba 3 (cu punctajul 6.2). • Ţara 10 are o singură medalie, de argint, la proba 3 (cu punctajul 6.1).

Timp maxim de execuţie/test: 1 secunde **Memorie totală disponibilă:** 8Mb/2Mb pentru stivă

Soluţie- Cohal Alexandru

Soluţia 1.1 – 100 puncte

Vom reţine cele mai bune trei punctaje de la fiecare probă sportivă în trei matrici: **tara[i][j]**, **intreg[i][j]** şi **zecimal[i][j]**, **i** fiind codul unei probe sportive, $1 \le i \le 500$ (500 fiind numărul maxim de probe sportive), iar j fiind indicele pentru poziţia pe podium, $1 \le j \le 3$.

- În matricea **tara** vom reţine codul ţării care a ocupat poziţia **j** la proba sportivă **i**;
- În matricea **intreg** vom reţine partea întreagă a punctajului sportivului din ţara **tara[i][j]** care a ocupat poziţia **j** la proba sportivă **i**;
- În matricea **zecimal** vom reţine partea zecimală a punctajului sportivului din ţara **tara[i][j]** care a ocupat poziţia **j** la proba sportivă **i**.

La fiecare citire a unei secvenţe, se actualizează cele trei matrici, avându-se în vedere următoarele cazuri:
- Dacă sportivul din secvenţa citită la momentul curent se află deja printre primii trei la proba sportivă la care participă, se actualizează punctajul său dacă acum a obţinut unul mai mare şi urcă în clasament dacă este cazul;
- Dacă sportivul din secvenţa citită la momentul curent **nu** se află deja printre primii trei la proba sportivă la care participă, se compară punctajul său cu punctajele primilor trei şi se actualizează clasamentul dacă este cazul.

Totodată, la citirea unei secvenţe se marchează în vectorul **okT** cu valoarea **1** ţara cu codul din secvenţă ca fiind participantă la Jocurile Olimpice, şi se contorizează în funcţie de acest vector numărul de ţări participante, **NrT**. La fel se face şi pentru probele sportive.
După crearea completă a clasamentului, se parcurg e matricea **tara** si se actualizeaza vectorii **aur[i]**, **argint[i]** şi **bronz[i]** care reţin câte medalii de aur, câte de argint, respectiv câte de bronz a obţinut ţara cu codul **i** la toate probele sportive.
La sfârşit se afişează valorile **NrT** şi **NrS** calculate anterior precum şi pentru fiecare ţară care are componenta din vectorul **okT** egală cu **1** codul acestei ţări urmat de componentele corespondente din vectorii aur, argint şi bronz.

Soluţia 1.2 – 100 puncte

Vom reţine cele mai bune trei punctaje de la fiecare probă într-o matrice de structuri având componentele **tara**, **intreg** şi **zecimal**.
Soluţia în continuare este la fel ca cea prezentată anterior.

Soluţia 2.1 – 100 puncte

Vom reţine punctajul (partea întreagă şi partea zecimală) ca un număr real. Vom reţine cele mai bune trei punctaje de la fiecare probă sportivă în două matrici: **tara[i][j]** şi **punctaj[i][j]**, **i** fiind codul unei probe sportive, $1 \le i \le 500$ (500 fiind numărul maxim de probe sportive), iar j fiind indicele pentru poziţia pe podium, $1 \le j \le 3$.
- În matricea **tara** vom reţine codul ţării care a ocupat poziţia **j** la proba sportivă **i**;
- În matricea **punctaj** vom reţine punctajul (ca număr real) sportivului din ţara **tara[i][j]** care a ocupat poziţia **j** la proba sportivă **i**.

Soluţia în continuare este la fel ca *Soluţia 1.1*.

Soluţia 2.2 – 100 puncte

Vom reţine cele mai bune trei punctaje de la fiecare probă într-o matrice de structuri având componentele **tara** si **punctaj**, **punctaj** fiind punctajul reţinut ca număr real.
Soluţia în continuare este la fel ca *Soluţia 1.2*.

```
/*Solutie 1.1
Autor: Alexandru Cohal*/
#include<fstream>
using namespace std;
#define SMAX 510
#define TMAX 510

int n;
int tara[SMAX][4],intreg[SMAX][4],zecimal[SMAX][4];
bool okT[TMAX],okS[SMAX];
int nrT,nrS;
int taraMax,probaMax;
int aur[TMAX],argint[TMAX],bronz[TMAX];
```

```
void updateTara(int x)
{
 taraMax = max(taraMax, x);

 if (okT[x] == 0)
      {
       okT[x] = 1;
       nrT ++;
      }
}

void updateProbaSport(int x)
{
 probaMax = max(probaMax, x);

 if (okS[x] == 0)
      {
       okS[x] = 1;
       nrS ++;
      }
}

void interschimbare(int lin, int col1, int col2)
{
 swap(tara[lin][col1], tara[lin][col2]);
 swap(intreg[lin][col1], intreg[lin][col2]);
 swap(zecimal[lin][col1], zecimal[lin][col2]);
}

void citire_actualizare()
{
 int i;
 int s,t,pi,pz;

 FILE *fin;
 fin = fopen("jo.in", "r");

 fscanf(fin, "%d", &n);
 for (i=1; i<=n; ++i)
      {
       fscanf(fin, "%d %d %d %d", &s,&t,&pi,&pz);

       updateTara(t);
       updateProbaSport(s);

       //daca sportivul din tara t se afla deja in clasament
       if (tara[s][1] == t) //daca ocupa pozitia 1
             {
             if (pi > intreg[s][1] || (pi == intreg[s][1] && pz > zecimal[s][1]))
                   {
                    intreg[s][1] = pi;
                    zecimal[s][1] = pz;
                   }
             }
       else
             if (tara[s][2] == t) //daca ocupa pozitia 2
                   {
                   if (pi > intreg[s][2] || (pi == intreg[s][2] && pz > zecimal[s][2]))
```

```
                        {
                         intreg[s][2] = pi;
                         zecimal[s][2] = pz;
                         if (intreg[s][2] > intreg[s][1] || (intreg[s][2] >=
intreg[s][1] && zecimal[s][2] > zecimal[s][1]))
                               interschimbare(s,1,2);
                        }
                  }
            else
                  if (tara[s][3] == t) //daca ocupa pozitia 3
                        {
                        if (pi > intreg[s][3] || (pi == intreg[s][3] && pz >
zecimal[s][3]))
                              {
                               intreg[s][3] = pi;
                               zecimal[s][3] = pz;
                               if (intreg[s][3] > intreg[s][2] || (intreg[s][3] >=
intreg[s][2] && zecimal[s][3] > zecimal[s][2]))
                                     {
                                      interschimbare(s,2,3);
                                      if (intreg[s][2] > intreg[s][1] || (intreg[s][2]
>= intreg[s][1] && zecimal[s][2] > zecimal[s][1]))
                                            interschimbare(s,1,2);
                                     }
                              }
                        }
                  else
                  //daca sportivul din tara t nu se afla deja in clasament
                       {
                        if (pi > intreg[s][3] || (pi >= intreg[s][3] && pz >
zecimal[s][3]))
                        //daca are un punctaj mai bun decat cel de pe pozitia 3
                              {
                               intreg[s][3] = pi;
                               zecimal[s][3] = pz;
                               tara[s][3] = t;
                               if (intreg[s][3] > intreg[s][2] || (intreg[s][3] >=
intreg[s][2] && zecimal[s][3] > zecimal[s][2]))
                               //daca are un punctaj mai bun decat cel de pe pozitia 2
                                     {
                                      interschimbare(s,2,3);
                                     if (intreg[s][2] > intreg[s][1] || (intreg[s][2]
>= intreg[s][1] && zecimal[s][2] > zecimal[s][1]))
                                      //daca are un punctaj mai bun decat cel de pe
pozitia 1
                                            interschimbare(s,1,2);
                                     }
                              }
                       }
      }

 fclose(fin);
}

void medalii()
{
 int i;
```

```
 //numar cate medalii de fiecare tip are fiecare tara
 for (i=1; i<=probaMax; ++i)
       if (okS[i] == 1)
             {
              aur[tara[i][1]]++;
              argint[tara[i][2]]++;
              bronz[tara[i][3]]++;
             }
}

void afisare()
{
 int i;

 FILE *fout;
 fout = fopen("jo.out", "w");

 fprintf(fout, "%d %d\n", nrS,nrT);
 for (i=1; i<=taraMax; ++i)
       if (okT[i] == 1)
             fprintf(fout, "%d %d %d %d\n", i,aur[i],argint[i],bronz[i]);

 fclose(fout);
}

int main()
{

 citire_actualizare();
 medalii();
 afisare();

 return 0;
}
```

Clasa a VII-a

KPuteri-Netedu Mircea Andrei **100 puncte**

Benvolio este elev in clasa a `VII`-a la Liceul de informatica „Grigore Moisil”. El a învăţat recent la informatică ce este un pătrat perfect şi să numere de câte ori apare un factor prim în descompunerea produsului primelor `N` numere naturale, dar cum nu a fost foarte atent la oră, profesorul i-a dat o temă specială. El a primit un număr `M` şi un şir `A` cu `N` numere naturale prime. Profesorul i-a cerut să numere, pentru fiecare număr `K` din şirul `A`, de câte ori apare în descompunerea în factori primi a produsului primelor `M` pătrate perfecte.
Benvolio nu ştie să rezolve o problemă atât de complexă, dar nu vrea nici să-ţi dezamăgească profesorul. El vă promite că, dacă îl veţi ajuta, vă va răsplăti cu `100` de puncte.

Cerinţă

Pentru un număr `M` şi un şir `A` de `N` numere naturale prime, se cere să se numere, pentru fiecare număr `K` din şirul `A` de câte ori apare el în descompunerea în factori primi a produsului primelor `M` pătrate perfecte.

Date de intrare

Pe prima linie a fişierului `kputeri.in` se vor afla numerele naturale `M` şi `N` printr-un spaţiu. Pe a doua linie se va a afla un şir `A` cu `N` numere naturale prime separate print-un spaţiu.

Date de ieşire

Fişierul de ieşire `kputeri.out` conţine o singura linie, pe care se va afla un şir de `M` numere, fiecare reprezentând răspunsul la cerinţa, pentru numărul prim corespunzător din şirul iniţial `M`.

Restricţii si Precizări

Numarul din sirul `A` nu sunt neaparat distincte.

`0< n,m < 200000, 0< k < 5000`

kputeri.in	kputeri.out	Explicaţii
5 3 2 3 5	6 2 2	Primele 5 pătrate perfecte sunt: 1, 4, 9, 16, 25. Descompunerea lor in factori prim este: 2^2, 3^2, 2^4, 5^2. $1*4*9*16*25=2^6*3^2*5^2$ Şirul M conţine 3 elemente: 2, 3, 5. 2 apare ca factor prim în descompunerea primelor 5 pătrate perfect de 6 ori. 3 apare ca factor prim în descompunerea primelor 5 pătrate perfect de 2 ori. 5 apare ca factor prim în descompunerea primelor 5 pătrate perfect de 2 ori.

Timp maxim de execuţie/test: 1 secundă **Memorie disponibilă:** 16Mb / 2 Mb pentru stivă

Soluţie- Netedu Mircea Andrei

$E=1^2 * 2^2 * 3^2 * 4^2 * ... * m^2 = (1*2*3*4*...*m)^2$

Notam cu P puterea maxima la care apare k in descompunerea produsului primelor m numere.

$P = (m / k) + (m / k^2) + … + (m / k^y)$, unde $y \in N^*$ cu proprietatea ca $k^y<=m$ si $k^{y+1}>m$.

$E= (k^P*x)^2 = k^{2*P} * x^2$, $x \in N^*$

Cum $(x,k)=1 \Rightarrow (x^2,k)=1 \Rightarrow$

=> Puterea la care apare K in descompunerea produsului primelor m patrate perfecte este 2*P.

```
#include<stdio.h>

long n,m,nr,i,k,a[6000];
int main()
{
	freopen("kputeri.in","r",stdin);
	freopen("kputeri.out","w",stdout);
	scanf("%ld %ld",&n,&m);
	for(i=1;i<=m;i++)
	{
		scanf("%ld",&k);
		if(a[k]==0)
		{
			nr=k;
			while(n/nr>0)
			{
				a[k]+=n/nr;
				nr+=nr;
			}
		}
		printf("%ld ",a[k]*2);
	}
	return 0;
}
```

Clasa a VII-a

Problema Pachete-Andrici Cezar Constantin **100 puncte**

Când îşi face bagajul pentru ONI, Porro îşi aminteşte că a promis colegilor că va aduce mai multe pachete de cărţi. Fratele lui fiind mai mic, a amestecat toate pachetele pe care le avea prin casă, chiar pierzând câteva cărţi.

Cerinţă

Fiind grăbit, vă cere ajutorul să-i spuneţi câte pachete complete are şi câte cărţi îi mai trebuie pentru a completa pachetele incomplete.

Date de intrare

Fişierul de intrare `pachete.in` va conţine două linii. Pe prima linie va fi numărul de cărţi pe care le-a găsit. Pe a doua linie vor fi scrise, separate de câte un spaţiu, cărţile găsite. Pentru fiecare carte va fi dată prin două caractere alăturate, primul reprezentând valoarea, iar al doilea culoarea cărţii respective.

Date de ieşire

Fiăierul de ieăire `pachete.out` va conţine, pe prima linie, numărul de cărţi care-i lipsesc pentru a nu lăsa nici un pachet incomplet. Pe a doua linie va fi numărul de pachete complete pe care le poate obţine.

Restricţii şi precizări

- Un pachet complet conţine toate combinaţiile posibile dintre `{2, 3, 4, 5, 6, 7, 8, 9, Z, J, Q, K, A}` şi `{I, R, F, T}`, `Z` codifică valoarea `10`.
- Un pachet incomplet este un pachet căruia îi lipsesc câteva cărţi.
- `1 ≤ numărul de cărţi ≤` 10^6

pachete.in	pachete.out	Explicaţii
53 2I 2I 2F 2R 2T 3I 3F 3R 3T 4I 4F 4R 4T 5I 5F 5R 5T 6I 6F 6R 6T 7I 7F 7R 7T 8I 8F 8R 8T 9I 9F 9R 9T ZI ZF ZR ZT JI JF JR JT QI QF QR QT KI KF KR KT AI AF AR AT	51 1	Porro poate completa un pachet complet, şi rămâne cu o singură carte, deci mai trebuie `51` de cărţi. În acest tabel, datorită spaţiului insuficient, cărţile nu apar scrise pe o singură linie, ca în fişierele de intrare/ieşire.

Timp maxim de execuţie/test: 0,7 secunde **Memorie disponibilă:** 2Mb / 1Mb pentru stivă

Soluţie-Andrici Cezar Constantin

Facem o matrice de frecvenţă cu 5 linii şi 15 coloane. a[i][j] = numărul de exemplare a cărţii care are culoarea i şi valoarea j (ji). Pentru valorile I, R, F, T vom folosi valorile 1, 2, 3, 4, iar pentru Z, J, Q, K, A vom folosi indicii 10, 11, 12, 13, 14. Se observă că numărul de pachete pe care le poate face e minimul din matrice. Numărul de cărţi care îi mai trebuie este = 52 * valoarea maximă din matrice - numărul de cărţi care se dau.

```
#include <cstdio>

int min, max, j, s, a[5][15], x, i, N;
char nr, cul;

int trsf(char c) {
    switch (c) {
        case 'J':
            return 11;
        break;
        case 'Q':
            return 12;
        break;
        case 'K':
            return 13;
        break;
        case 'A':
            return 14;
        break;
        case 'I':
            return 1;
        break;
        case 'R':
            return 2;
        break;
        case 'F':
            return 3;
        break;
        case 'T':
            return 4;
```

```
        break;
    }

    return c -  '0';
}

int main() {
       freopen("pachet.in","r",stdin);
       freopen("pachet.out","w",stdout);

            scanf("%d\n", &N);

            for (i = 1; i <= N; ++i) {
                scanf("%c%c ", &nr, &cul);
                x = ++a[trsf(cul)][trsf(nr)];
                max = (max < x)?x:max;
            }

            for (i = 1; i <= 4; ++i) {
                for (j = 1; j <= 14; ++j) {
                    min = (min > a[i][j])?a[i][j]:min;
                }
            }

            printf("%d\n%d\n", max*52-N, min);

       return 0;
}
```

Clasa a VIII-a

Char-virus-Negruş Ştefan 100 puncte

Char-virus este un virus care transformă şirurile de litere mici, făcând înlocuiri impuse de un alfabet virusat. Un **alfabet virusat** are n litere distincte $v_1v_2...v_n$, ordonate astfel încât un caracter v_i are puterea de substituţie mai mare decât caracterele $v_1v_2...v_{i-1}$. Pe un şir de litere din acelaşi alfabet, Char-virus acţionează astfel: într-o secundă, parcurge şirul de la stânga la dreapta şi crează un nou şir temporar în care fiecare caracter este egal cu cel din şirul original, dacă vecinii săi au putere de substituţie mai mică sau egală decât a lui, în caz contrar este egal cu caracterul vecin cu puterea de substituţie cea mai mare. Şirul original este înlocuit de cel temporar şi va fi transformat în continuare. După un număr de secunde în care Char-virus acţionează repetat, şirul va deveni **uni-virusat,** adică va fi format din caractere identice.

Cerinţa

Să se determine pentru fiecare dintre cele m şiruri virusate date, numărul minim de secunde după care acesta devine **uni-virusat**.

Date de intrare

Pe prima linie a fişierului `virus.in` se află două numere naturale nenule `n` şi `m` cu semnificaţiile din enunţ. Pe următoarea linie sunt date cele `n` caractere $v_1v_2v_3...v_n$ ale alfabetului virusat în ordinea crescătoare a puterii lor de substituţie.

Fiecare dintre următoarele `m` linii are forma:

$K\ c_1c_2...c_K$

unde K=lungimea şirului virusat iar $c_1c_2...c_K$ literele din care este format. Şirurile sunt formate doar din litere mici, nu neapărat distincte, ale alfabetului virusat.

Date de ieşire

Fişierul de ieşire `virus.out` va conţine m linii, pe fiecare linie fiind scris numărul minim de secunde necesare transformării şirului corespunzător într-un şir **uni-virusat**.

Restricţii

$1 \le n \le 26;\ 1 \le m \le 40;\ 1 \le K \le 25000$

Pentru 50% din teste $1 \le K \le 128$.

virus.in	virus.out	Explicaţii
6 4 bcaqxt 9 abbbaaaba 7 cbqtbbc 4 qtcq 6 bbbbbb	2 3 2 0	Şirul abbbaaaba se modifică astfel: Secunda 1: aabaaaaaa, pentru ca a are puterea de substituţie mai mare decât b (se află după b în alfabetul virusat); Secunda 2: aaaaaaaaa, şirul devenind uni-virusat. Şirul cbqtbbc se modifică astfel: Secunda 1: cqtttcc Secunda 2: qtttttc Secunda 3: ttttttt Şirul qtcq se modifică astfel: Secunda 1: tttq Secunda 2: tttt Şirul bbbbbb este deja uni-virusat.

Timp maxim de execuţie/test: 0,1 secundă **Memorie disponibilă:** 4Mb / 1Mb pentru stivă

Soluţie-Negruş Ştefan

Soluţie 50 puncte **O (N * K^2)**

Pentru fiecare şir de caractere se crează un şir nou în care se modifică fiecare caracter in funcţie de caracterele vecine în şirul iniţial, conform regulii descrise in enunţ. Iar în urma modificărilor şirul vechi este înlocuit de şirul nou. O mică optimizare se poate face prin crearea soluţiei in felul următor :

Fie A[] şirul iniţial, fie B[] şirul nou format, în această etapă ar fi necesară copierea caracterelor din şirului B în şirul A, însă situaţia următoare se poate construi direct din şirul B în şirul A. Etapele fiind:

A –> B
B –> A
A –> B
.
.
cât timp se produc modificări

Soluţie 100 puncte **O (N * K)**

Se poate observa ca în fiecare şir va exista un caracter care are cea mai mare putere de subtituţie astfel că acesta va fi unicul caracter din şirul uni-virusat detereminat de şirul iniţial. Datorită acestui lucru putem sa ţinem cont doar de poziţiile acelui caracter şi să verificăm doar distanţele dintre apariţiile acestui caracter.

Considerăm şirurile indexate de la 0 şi fie ‘Z’ caracterul cu puterea de substituţie cea mai mare şi avem exemplul (caracterul ‘.’ înlocuieşte celelalte caractere, în sensul ca nu ţinem cont de acestea):

|……..Z……..ZZ……Z……ZZZZZZZ……..Z……|
|…Ds...Z……Di....ZZ…..Di.. Z……Di..……ZZZZZZZ……Di....Z……..Df..……|

Ds = distanţa de la început pana la primul caracter = poziţia primului caracter
Df = distanţa de la ultima apariţie a caracterului ‘Z’ pâna la sfârşitul şirului
Di (i = 1..t) = distanţa intre două apariţii ale caracterului ‘Z’, însă vom considera această distanţă/2 deoarece la fiecare pas această diferenţă scade cu 2, ambele caractere substituie caracterele alăturate în acelaşi timp.

În concluzie valoarea căutată pentru fiecare şir = max (Ds , Df , Di) , cu i = 1..t , unde t = numărul de distanţe.

```
/* Negrus Stefan - virus - 100 */
#include<fstream>

#define INfile "virus.in"
#define OUTfile "virus.out"

#define NMAX 25009

using namespace std ;
```

```
ifstream F ( INfile ) ;
ofstream G ( OUTfile ) ;

int N , M ;

char V [ 27 ] ;
char S [ NMAX ] ;
int val [ 27 ] ;

void read () ;
void solve () ;

int main ()
{
      int i ;
      read () ;
      for ( i = 1 ; i <= M ; ++ i )
            solve () ;
      G.close () ;
      F.close () ;
      return 0 ;
}

void read ()
{
      int i ;
      F >> N >> M ;
      F.get () ;
      F.getline ( V , 30 ) ;
      for ( i = 0 ; i < N ; ++ i )
            val [ V [ i ] - 'a' ] = i ;
}

void solve ()
{
      int K ;
      int pozl , pozf ,i , lmax , Z , pozi , dist ;
      F >> K >> S ;
      Z = -1 ;
      lmax = -1 ;
      pozl = 0 ;
      for ( i = 0 ; i < K ; ++i )
      {
            if ( val [ S [ i ] - 'a' ] > Z )
            {
                  lmax = i ;
                  pozl = i ;
                  Z = val [ S [ i ] - 'a' ] ;
            }
            else if ( val [ S [ i ] - 'a' ] == Z )
            {
                  dist =  ( i - pozl ) >>1 ;
                  if (  dist  > lmax  )
                        lmax = dist ;
                  pozl = i ;
            }
      }
      dist = K - pozl - 1 ;
      if ( dist > lmax )
            lmax = dist ;
      G << lmax << '\n' ;
}
```

Clasa a VIII-a

Materii-Filimon Marta Diana şi Cohal Alexandru 100 puncte

Ruxandra a mers să se înscrie la facultate împreuna cu fraţii ei mai mici Alf şi Zilla. Aceştia au remarcat că vor fi nevoiţi să aştepte o vreme, aşa că au inventat un joc inspirat de foaia matricolă a surorii lor.
Foaia matricolă a Ruxandrei conţine un tabel cu `n` linii şi `m` coloane completat cu punctajele obţinute de ea în timpul liceului, în ordine cronologică, fiecare linie reprezentând câte o materie.
Jocul inventat are *definiţii* şi urmează doi *paşi*.

Definiţii:

1) O materie are ***proprietatea 1*** dacă scriind punctajele unul lângă altul obţinem un număr palindrom.
2) O materie are ***proprietatea 2*** dacă suma punctajelor obţinute într-o perioadă de timp este divizibilă cu `k`, dat.
3) O materie are ***proprietatea 3*** dacă are ***proprietăţile 1 şi 2***.

Paşii jocului:

1) Se elimină toate materiile ce au *proprietatea 3*.
2) Oricare două materii consecutive, în care prima materie are *proprietatea 1* şi a doua materie are *proprietatea 2,* se elimină.

Cerinţă

Ajutaţi-i pe Alf şi Zilla să determine:

a) numărul de materii care au doar *proprietatea 1*
b) numărul de materii care au doar *proprietatea 2*
c) numărul de materii care au *proprietatea 3*
d) numărul de materii care *nu* sunt *eliminate*.

Date de intrare

Pe prima linie a fişierului `materii.in` se vor afla două numere întregi separate printr-un singur spaţiu, `n` şi `m` reprezentând numărul de materii, respectiv numărul de punctaje pe care le-a obţinut Ruxandra la fiecare materie. Pe a doua linie a fişierului se va afla numărul natural `k`, cu semnificaţia din enunţ, iar pe următoarele `n` linii se vor afla câte `m` numere naturale separate prin câte un spaţiu, reprezentând punctajele din foaia matricolă a Ruxandrei.

Date de ieşire

Fişierul `materii.out` va conţine patru numere separate prin câte un spaţiu, reprezentând răspunsurile la cele patru cerinţe în ordinea: cerinţa `a, b, c, d`.

Restricţii şi precizări

$4 \le n, m, k \le 2000$
Pentru `50%` din teste $m \le 100$.
Pentru alte `20%` din teste $k \le m$.
Punctajele sunt numere întregi cuprinse în intervalul `[1,10000]`.
Dacă o materie este ***eliminată*** atunci linia corespunzătoare ei dispare din foaia matricolă adică toate punctajele materiilor care sunt după aceasta, urcă cu o linie mai sus.
Un număr este ***palindrom*** dacă citit de la dreapta la stânga este egal cu numărul citit de la stânga la dreapta.
Se va acorda `10%` din punctaj pentru cerinţa **a**, `20%` din punctaj pentru cerinţa **b**, `10%` din punctaj pentru cerinţa **c**, iar restul de `60%` se va acorda pentru rezolvarea tuturor celor patru cerinţe.

Exemplu

materii.in	**materii.out**	**explicaţii**
5 5 10 7 9 11 9 7 3 3 3 3 3 7 21 5 4 19 5 9 3 8 6 5 53 8 8 355	2 2 1 2	Materia 1 are proprietatea 3 (`791197` este palindrom şi (`1+9):10=2rest0`). Materia 2 are proprietatea 1 (`33333` este palindrom). Materia 3 are proprietatea 2 (`(21+5+9):10=3rest0`). Materia 4 are proprietatea 2 (`(9+3+8):10=2rest0`). Materia 5 are proprietatea 1 (`55388355` este palindrom).

Timp maxim de execuţie/test: 3 secunde **Memorie disponibilă:** 16Mb / 2 Mb pentru stivă

Soluţie-Cohal Alexandru şi Filimon Marta

Pentru a verifica dacă o materie are ***proprietatea 1*** concatenăm pe rând toate punctajele de la fiecare materie într-un şir de caractere, iar apoi verificăm dacă acest şir de caractere este palindrom.

Pentru a verifica dacă o materie are ***proprietatea 2*** aplică *principiul lui Dirichlet* astfel: Întâi calculăm pentru fiecare punctaj de la o materie suma tuturor punctajelor până la punctajul curent, inclusiv punctajul curent. Dacă una dintre aceste sume este divizibilă cu numărul `k`, atunci materia are ***proprietatea 2***. Dacă nu avem nicio astfel de sumă, căutăm dacă există două dintre aceste sume care sa aibă acelaşi rest la împărţirea la numărul `k`. Dacă am găsit două astfel de sume, atunci materia are ***proprietatea 2***.

Dacă o materie are atât ***proprietatea 1*** cât şi ***proprietatea 2***, atunci ea are ***proprietatea 3***.

Actualizăm numărul de materii care au ***proprietatea 1***, ***proprietatea 2*** sau ***proprietatea 3***.

Dacă o materie are ***proprietatea 1***, ***proprietatea 2*** sau nicio proprietate, adăugăm numărul proprietăţii (`1`, `2` sau `0` (pentru materia cu nicio proprietate)) într-o stivă. Atunci când avem în stivă o secvenţa formată dintr-un `1` şi un `2` în această ordine, eliminăm aceste două elemente şi continuăm verificarea. Numărul de elemente care rămân în stivă reprezintă numărul de materii care nu sunt eliminate.

```
#include<fstream>
#include<cstdlib>
#include<cstring>
#define NMAX 2012

using namespace std;

ifstream f("materii.in");
ofstream g("materii.out");

int a[NMAX], fr[NMAX], st[NMAX], nst=0, nr1, nr2, nr3, n,

m, k;
char c[4*NMAX], nr[10];

int tip2()
{
      int i, ok=0;

      for (i=1; i<k; ++i)
      {
            if (fr[i]>1) ok=1;
            fr[i]=0;
      }

      if (fr[0]>0) {ok=1;fr[0]=0;}

      return ok;
}

int tip1()
{
      int p=0, u=strlen(c)-1, ok=1;

      while (p<u)
      {
            if (c[p]!=c[u]) {ok=0; break;}
            ++p;
            --u;
      }

      return ok;
}
```

```
void Solve()
{
      int i, j, s=0, tip;

      f>>n>>m>>k;

      for (i=1; i<=n; ++i)
      {
            for (j=1; j<=m; ++j)
            {
                  f>>a[j];
                  s=(s+a[j])%k; ++fr[s];
                  itoa(a[j], nr, 10);
                  strcat(c, nr);
            }
            tip=0;

            if (tip1()) ++tip;
            if (tip2()) tip+=2;

            if (tip==1)
            {
                  ++nr1;
                  st[++nst]=1;
            }
            if (tip==2)
            {
                  ++nr2;
                  if (st[nst]==1) st[nst--]=0;
                  else st[++nst]=2;
            }
            if (tip==0) st[++nst]=0;
            if (tip==3) ++nr3;

            s=0;c[0]='\0';
      }
      g<<nr1<<" "<<nr2<<" "<<nr3<<" "<<nst<<"\n";
}

int main()
{
      Solve();

      f.close();
      g.close();
      return 0;
}
```

Clasa a IX-a

Spirala-Stoian Vlad

100 puncte

Raţuşca Ducky este o raţuşcă ametiţă. Ei îi place foarte mult să se învârtă în spirală şi să mănânce ouă din ciocolată. Într-o zi s-a gândit ca la ora mesei să îşi pună în spirală, începând din mijloc şi apoi în sensul acelor de ceasornic (în imaginea alăturată este exemplificată o spirala `5x5`), la fiecare pas, atâtea ouă câţi paşi îi va lua să ajungă până acolo. Fiind ameţită, ea îşi aminteşte să mănânce ouăle doar când ajunge pe una dintre cele 2 diagonale.

21	22	23	24	25
20	7	8	9	10
19	6	1	2	11
18	5	4	3	12
17	16	15	14	13

Cerinţă

Având ca intrare lungimea `N` a laturii spiralei, Ducky vă roagă să-i spuneţi câte ouă de ciocolată îşi va aminti să mănânce dacă trece prin toată spirala.

Date de intrare
Din fişierul de intrare `spirala.in` se citeşte numărul natural `N` impar.

Date de ieşire
În fişierul `spirala.out` se va scrie numărul total `T` de ouă pe care le-a mâncat răţuşca `(modulo 2 223 333 667)`.

Restricţii
`1 < N < 1 000 000, N` este număr impar

spirala.in	**spirala.out**
5	101
3	25

Soluţie- Stoian Vlad

Se observa ca numerele cautate se gasesc dupa urmatorul model: incepem cu 1 dupa care adaugam 2 de 4 ori, apoi 4 de 4 ori, 6 de 4 ori, 8 de 4 ori, 2k de 4 ori. (1, 3, 5, 7, 9, 13, 17, 21, 25, 31, 37, 43…)
Pentru retinerea sumei trebuie folosita o variabila de tip unsigned int (sau long long). In caz ca e unsigned, trebuie atentie la depasiri. In caz ca suma depaseste unsigned int, trebuie scazuta mai intai valoare modulo-ului.

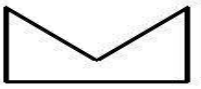
```
#include <stdio.h>

/*
 * PROBLEMA: Spirala
 *
 * AUTOR: Vlad Stoian
 *
 */

unsigned int mod_value = 2223333667;

unsigned int get_diag_sum(unsigned int n)
{

      unsigned int sum = 1;
      unsigned int count = 1;
      for (unsigned int i = 1; i <= (n>>1); i++)
      {
            for (unsigned int j = 0; j < 4; j++)
            {
                  count = (count + (i*2)) % mod_value;
                  if ( (sum + count) < sum)
                  {
                        sum -= mod_value;
                  }
                  sum = (sum + count) % mod_value;

            }
      }

      return sum;
}

int main()
{

      FILE *fin, *fout;

      fin = fopen("spirala.in", "rt");
      int n;
      fscanf(fin, "%d", &n);
      fclose(fin);
```

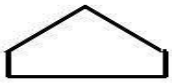
```
    fout = fopen("spirala.out", "wt");
    unsigned int sum = get_diag_sum(n);
    fprintf(fout, "%u\n", sum);
    fclose(fout);

    return 0;
}
```

Problemele de concurs 2011
Clasa a V-a

Zid – Onuţu Codrin 100 puncte

Căpitanul din garda Regelui Arthur este responsabil de protejarea cetăţii. El are un număr de **N** oameni pentru a-i pune în poziţii în spatele zidului. Ştiind înălţimea zidului în poziţii, dar şi aspectul lui (aspect de **vale** sau de **munte**) şi faptul că **zidul este simetric faţă de mijloc**, să se determine un mod de a aranja soldaţii pentru a apăra cetatea astfel încât ei să aibă vizibilitate. Un soldat are vizibilitate dacă este cu cel puţin 20 de centimetri mai înalt decât zidul din faţa sa.


aspect de vale


aspect de munte

Cerinţă
Determinaţi un mod de a aranja corect toţi soldaţii.

Date de intrare
Pe prima linie se vor afla numerele **N** de poziţii şi un număr întreg **x** cu valoarea 1 dacă zidul are aspect de vale sau 2 dacă zidul are aspect de munte. Pe următoarea linie se găsesc **N** valori reprezentând înălţimile soldaţilor, iar pe a treia linie vor fi **N** valori reprezentând înălţimea zidului în poziţiile în care căpitanul vrea să-şi pună soldaţii.

Date de ieşire
Pe prima linie se va afişa amplasarea soldaţilor de-a lungul zidului. Se va afişa −1 dacă nu există un mod corect de a aranja soldaţii. Este acceptată orice soluţie ce îndeplineşte condiţiile de mai sus.

Restricţii
- 3≤**N**≤100.
- 120≤înălţimile poziţiilor≤200
- 150≤înălţimile soldaţilor≤220

zid.in	zid.out	Explicaţii
5 2 200 176 189 173 190 150 160 180 160 150	173 189 200 190 176	Zidul are aspect de munte. În fiecare poziţie diferenţa dintre înălţimea soldatului şi înălţimea zidului este mai mare sau egal decât 20. (173-150=23, 189-160=29, 200-180=20, 190-160=30, 176-150=26)
6 1 200 180 180 175 199 173 170 160 150 150 160 170	200 180 175 173 180 199	Zidul are aspect de vale. În fiecare poziţie diferenţa dintre înălţimea soldatului şi înălţimea zidului este mai mare sau egal decât 20. (200-170=30, 180-160=20, 175-150=25, 173-150=23, 180-160=20, 199-170=29)

Timp maxim de execuţie/test: 1 secundă.

Soluţie - Onuţu Codrin

```
#include <fstream>
using namespace std;
ifstream in("zid.in");
ofstream out("zid.out");
int n, a[100],j,aux, b[100],c[100],i,aspect,posi,posf,k;
int main()
{
   in>>n>>aspect;
```

```
    k=0;
    for(i=0;i<n;i++)
        in>>a[i];
    for(j=0;j<n;j++)
        in>>b[j];
    if(aspect==1)
    {
        for(i=0;i<n;i++)
            for(j=i+1;j<n;j++)
                if(a[i]<a[j]) {aux=a[i];a[i]=a[j];a[j]=aux;}
        posi=0;
        posf=n-1;
        for(i=0;i<n;i++)
        {
            if(i%2==0) {c[posi]=a[i];posi++;}
            if(i%2==1) {c[posf]=a[i];posf--;}
        }
        for(i=0;i<n;i++)
            if(!(c[i]-20>=b[i])) {k=1;break;}
    }
    else
    {
        for(i=0;i<n&&k==0;i++)
            for(j=i+1;j<n;j++)
                if(a[i]>a[j]) {aux=a[i];a[i]=a[j];a[j]=aux;}
        posi=0;
        posf=n-1;
        for(i=0;i<n;i++)
        {
            if(i%2==0) {c[posi]=a[i];posi++;}
            if(i%2==1) {c[posf]=a[i];posf--;}
        }
        for(i=0;i<n;i++)
            if(!(c[i]-20>=b[i])) {k=1;break;}
    }
    if(k==0)
        for(i=0;i<n;i++)
            out<<c[i]<<' ';
    else out<<"-1";
    out<<'\n';
    in.close();
    out.close();
    return 0;
}
```

Soluţie - Filimon Marta

```
program p1;
type vect=array[1..100] of integer;
var f, g:text;
    a, b:vect;
    i, n, op, ok, u:integer;

procedure ordoneaza(var a:vect);
var i, j, aux:integer;
begin
     for i:=1 to n-1 do
         for j:=i+1 to n do
             if (a[i]>a[j]) then begin aux:=a[i]; a[i]:=a[j]; a[j]:=aux;end;
end;
```

```
begin
     assign(f,'zid.in');reset(f);
     assign(g,'zid.out');rewrite(g);

     read(f, n, op);
     for i:=1 to n do read(f, a[i]);
     for i:=1 to n do read(f, b[i]);

     ordoneaza(a);
     ordoneaza(b);

     ok:=1;
     for i:=1 to n do
         if (a[i]-b[i]<20) then begin ok:=0; break; end;

     if (ok=0) then writeln(g,'-1')
     else
     if (op=1) then
     begin
          u:=n;
          while (u>0) do
          begin
               write(g, a[u], ' ');
               u:=u-2;
          end;
          if (n mod 2=1) then u:=2
          else u:=1;
          while (u<=n) do
          begin
               write(g, a[u], ' ');
               u:=u+2;
          end;
     end
     else
     begin
          u:=1;
          while (u<=n) do
          begin
               write(g, a[u], ' ');
               u:=u+2;
          end;
          if (n mod 2=1) then u:=n-1
          else u:=n;
          while (u>0) do
          begin
               write(g, a[u], ' ');
               u:=u-2;
          end;
     end;
     close(f);
     close(g);
end.
```

Clasele a V-a şi a VII-a

Bombo - Ţucăr Liana **100 puncte**

De ziua ei, Măgduţa are invitaţi `N` copii pe care îi serveşte cu bomboane de `N` tipuri.Îşi aşează invitaţii în cerc, în ordine, de la copilul `1` la copilul `N` şi le împarte bomboane după următoarea regulă:

- Primul copil primeşte o bomboană de tipul `1`
- Al doilea copil primeşte două bomboane de tipul `2`
- Al treilea copil primeşte trei bomboane de tipul `3`

- .....
- Al N -lea copil primeşte N bomboane de tipul N
- Primul copil primeşte o bomboană de tipul 1
- .....şi aşa mai departe până când ajunge la un copil căruia nu îi mai poate da bomboane.

Cerinţă

Ajutaţi-o pe Măgduţa să afle copilul care a primit cele mai multe bomboane şi numărul bomboanelor primite de acesta. Dacă sunt mai mulţi copii cu acelaşi număr maxim de bomboane, Măgduţa vrea să-l afle pe cel cu numărul de ordine minim.

Date de intrare

Pe prima linie a fişierului bombo.in se găseşte numărul de copii, N. Pe următoarea linie se găsesc N numere naturale, a_1 a_2 a_3... a_N, separate prin câte un spaţiu, fiecare a_i reprezentând numărul de bomboane de tipul i pe care le are iniţial Măgduţa.

Date de ieşire

Pe prima linie a fişierului bombo.out se găsesc două numere a b, unde a este copilul care a primit cele mai multe bomboane iar b, numărul bomboanelor primite de acesta .

Restricţii şi precizări

- $2 \le N \le 50$
- $N \le a_1+a_2+a_3+.. +a_N \le 500\ 000$
- Există cel puţin câte o bomboană din fiecare tip.

bombo.in	bombo.out	Explicaţie
3 6 5 7	3 6	**Copil** **Nr. Bomboane primite la fiecare pas** 1 1 2 2 3 3 1 1 2 2 3 3 1 1 Nu se mai pot împărţi deoarece nu a mai rămas decat o bomboană de tipul 2. În total primul copil a primit 3 bomboane, al doilea a primit 4, iar al treilea 6.

Timp maxim de execuţie/test: 1 secundă.

Soluţie - Ţucăr Liana

```
#include <stdio.h>
#define nmax 1000
long b, n, i, x, nt, poz, v[nmax], nb, crezmax, rezmax;
bool ok;
int main()
{
      freopen("bombo.in","r",stdin);
      freopen("bombo.out","w",stdout);
      scanf("%ld %ld",&b, &n);
      for (i=1;i<=n;i++)
            scanf("%ld",&v[i]);
      nt=b;
      for (i=1;i<=n;i++)
            if (v[i]/i<nt)
                  nt=v[i]/i;
      poz=1;      ok=1;
      while (ok)
      {
            nb=poz*nt;
            if (v[poz]-nb>=poz)
            {
                  nb+=poz;
```

```
                rezmax=nb; crezmax=poz;
            }
            else
                break;
            poz++;
        }
        if (n*nt>rezmax)
        {    rezmax=n*nt;    crezmax=n; }
        printf("%ld %ld",crezmax, rezmax);
        return 0;
}
```

Soluţie – Banu Denis

```
program Bomboane;
var v,v2:array [1..50] of longint;
    n,i,nrmax,max,ok:longint;
    f,g:text;
begin
     assign(f,'bombo.in');
     reset(f);
     assign(g,'bombo.out');
     rewrite(g);
     read(f,n);
     for i:=1 to n do
         read(f,v[i]);
     i:=1;
     while (v[i]+i>i) and (ok=0) do
           begin
                v[i]:=v[i]-i;
                if v[i]+i>=i then
                v2[i]:=v2[i]+i;
                if v[i]+i<i then ok:=1;
                if i=n then i:=0;
                i:=i+1;
           end;
     max:=v2[1];nrmax:=1;
     for i:=2 to n do
         begin
              if max<v2[i] then
                 begin
                      max:=v2[i];
                      nrmax:=i;
                 end;
         end;
     write(g,nrmax,' ',max);
     close(f);
     close(g);
     end.
```

Clasele a VI-a şi a VII-a

Robot – Filimon Marta **100 puncte**

Fratele Ruxandrei este foarte priceput la electronică. El construieşte `k` roboţi şi îi aşează pe un caroiaj care are maxim 10^9 linii şi 10^9 coloane, în care numerotarea liniilor şi coloanelor începe de la `1`. Ruxandra crează pentru fiecare robot câte o succesiune de `nr` comenzi. Există `4` tipuri de comezi: comanda de tipul `1` pentru care robotul se deplasează la nord, comanda de tipul `2` pentru care robotul se deplasează la est, comanda de tipul `3` pentru care robotul se deplasează la sud şi comanda de tipul `4` pentru care robotul se deplasează la vest. Ruxandra nu ştie însă că, atunci când `2` sau mai mulţi roboţi se întâlnesc simultan în acelaşi pătrăţel, aceştia dispar de pe caroiaj ca prin magie.

Cerinţă

Să se determine numărul de roboţi ce se vor afla pe caroiaj la finalul executării tuturor comenzilor.

Date de intrare

Pe prima linie a fişierului robot.in se află numerele k şi nr, cu semnificaţiile din enunţ. Pe următoarele k linii se află, pentru fiecare robot: poziţia lui de start sub forma unei perechi **linie coloană** apoi nr numere reprezentând mişcările sale.

Date de ieşire

Se va afişa în fişierul robot.out un singur număr reprezentând numărul de roboţi care rămân pe caroiaj.

Restricţii şi precizări

- $2 \le k, n \le 10$
- **Un robot nu va părăsi caroiajul după succesiunea de mişcări dată.**

robot.in	robot.out	Explicaţii
4 5 2 3 2 3 2 3 4 3 1 3 2 2 2 2 1 4 3 2 2 2 2 4 1 3 3 3 3 3	2	

	1	2	3	4
1				R3
2			R1	
3	R2			
4	R4			

Fiecare din cei 4 roboţi execută 5 comenzi. Roboţii 1 si 3 se ciocnesc în pătrăţelul de pe linia 2, coloana 4 şi dispar. Ceilalţi doi roboţi îşi continuă drumul.

Robot/ moment	1	2	3	4	5
1	2,4	-	-	-	-
2	4,1	4,2	4,3	4,4	4,5
3	2,4	-	-	-	-
4	4,1	5,1	6,1	7,1	8,1

Timp maxim de execuţie/test: 1 secundă.

Soluţie-Filimon Marta

```
program p1;

type robot=record
                 i,j:longint;
                 vz:shortint;
                 d:array[1..100] of shortint;
           end;

var f,g:text;
    a:array[1..100] of robot;
    di:array[1..4] of integer;
    dj:array[1..4] of integer;
    n,m,k,nr:integer;

procedure Directii;
begin
     {nord}di[1]:=-1; dj[1]:=0;
     {est}di[2]:=0; dj[2]:=1;
     {sud}di[3]:=1; dj[3]:=0;
     {vest}di[4]:=0; dj[4]:=-1;
end;

procedure Citeste;
var i,j:integer;
begin
     read(f,k,nr);
     for i:=1 to k do
     begin
          read(f, a[i].i, a[i].j);
          for j:=1 to nr do read(f, a[i].d[j]);
     end;
end;

procedure Verifica(p:integer);
var i,ok:integer;
```

```
begin
     ok:=0;
     for i:=p+1 to k do
         if (a[p].i=a[i].i) and (a[p].j=a[i].j) and (a[i].vz=0) then
         begin
              ok:=1;
              a[i].vz:=1;
         end;
     a[p].vz:=ok;
end;

procedure Modifica(t:integer);
var i:integer;
begin
     for i:=1 to k do
     begin
          a[i].i:=a[i].i+di[a[i].d[t]];
          a[i].j:=a[i].j+dj[a[i].d[t]];
     end;
end;

procedure Simuleaza;
var i,t:integer;
begin
     for t:=1 to nr do
     begin
          Modifica(t);
          for i:=1 to k do
              if a[i].vz=0 then Verifica(i);
     end;
end;

procedure Scrie;
var nr, i:integer;
begin
     nr:=0;
     for i:=1 to k do
         if (a[i].vz=0) then nr:=nr+1;
     writeln(g, nr);
end;

begin
     assign(f,'robot.in');reset(f);
     assign(g,'robot.out');rewrite(g);

     Directii;

     Citeste;

     Simuleaza;

     Scrie;

     close(f);
     close(g);
end.
```

Clasa a VII-a

ILIS – Negruş Ştefan **100 puncte**

O navă extraterestră de pe planeta ILIS s-a prăbuşit pe Terra. Cercetând aparatura navei, s-a descoperit un calculator de buzunar mai ciudat. El face operaţiile noastre matematice numai că pe tastatură, în locul cifrelor, apar literele mici ale alfabetului englez. Astfel oamenii de ştiinţă şi-au dat seama din calcule că extratereştrii lucrează în baza `26`, iar fiecare literă reprezintă o cifră în această bază, astfel: `a=0, b=1 … z=25`.
Folosind acest calculator oamenii de ştiinţă au efectuat următoarele calcule:

c **y+** **ba**	**bbc** **zzz+** **bbbb**	**baa** **bac** **bb+** **cbd**

Cerinţă
Voi sunteţi angajaţi să scrieţi un program care să facă adunările precum calculatorul original.

Date de intrare
În fişierul de intrare pe prima linie va fi `N`, numărul de şiruri ce trebuie adunate, apoi `N` linii, pe linia `i+1` va fi al `i`-lea şir. Fiecare linie se va termina cu un caracter enter (sfârşit de linie) .

Date de ieşire
Pe singura linie din fişierul de ieşire va fi rezultatul adunării celor `N` şiruri, urmat de caracterul **enter (sfârşit de linie)**.

Restricţii şi precizări
- `1<`**`N`**`≤10`
- `0<`lungimea şirurilor`≤20 000`
- Şirurile nu vor începe cu un caracter `'a'( 0 )`

ilis.in	**ilis.out**	**Explicaţii**
3 dwa yac bb	Bbxd	Şirurile din fişierul de intrare sunt echivalente cu următoarea adunare în baza 26: **3 22 0** **24 0 2** **1 1 +** **1 1 23 3** **b b x d**

Timp maxim de execuţie/test: 1 secundă.

Soluţie – Negruş Ştefan

```
#include<fstream.h>
#include<string.h>

ifstream f("ilis.in");
ofstream g("ilis.out");

int N,lgmax;
char x,a[20001],SI[20001];


void mirror (char *a);
void read();
void solve1();
void write();
void add(char *A, char *B);

int main()
{
   read();
   solve1();
```

```
   write();

   f.close();
   g.close();
   return 0;
}

void read()
{
   f>>N;
   f.get();
}

void solve1()
{
   int i;
   f.getline(SI,20000);
   mirror(SI);
   for (i=2;i<=N;++i)
   {
      f.getline(a,20001);
      mirror(a);
      add(SI,a);
   }
   mirror(SI);
}

void add(char *A, char *B)
{
      int i, t = 0,lg1=strlen(A),lg2=strlen(B),x;
      int minn=(lg1>lg2?lg2:lg1);
      for (i=0;i<minn;++i)
      {
         x=(A[i]-'a')+t+(B[i]-'a');
         //A[i]='a'+(A[i]-'a')+t+(B[i]-'a');
         t=x/26;
         A[i]='a'+x%26;
      }

      if (lg1<lg2)
         for (i=lg1;i<lg2;++i)
         {
            x=t+(B[i]-'a');
            //A[i]=t+B[i];
            t=x/26;
            A[i]='a'+x%26;
         }

      while (t)
      {
         if (i>=lg1)
            A[i]='a'+t;
         else A[i]+=t;
         t=(A[i]-'a')/26;
         //A[i]='a'+(A[i]-'a')%26;
         ++i;
      }

      if (i>lg1) A[i]='\0';
}
```

```
void mirror (char *a)
{
   int i=0,lg=strlen(a)-1;
   char aux;
   while (i<lg)
   {
      aux=a[i];
      a[i]=a[lg];
      a[lg]=aux;
      ++i;
      --lg;
   }
}

void write()
{
   g<<SI<<'\n';
}
```

Soluţie – Filimon Marta

```
program p1;
var f,g:text;
    s, a: array[0..20100] of integer;
    ns, na, i, n:longint;

procedure Citeste;
var c:char;
    p, u, aux:integer;
begin
      na:=0;
      while not eoln(f) do
      begin
           read(f,c);
           na:=na+1;
           a[na]:=ord(c)-ord('a');
      end;
      p:=1; u:=na; a[0]:=na;
      while (p<u) do
      begin
           aux:=a[p]; a[p]:=a[u]; a[u]:=aux;
           p:=p+1; u:=u-1;
      end;
      readln(f);
end;

procedure Suma;
var i, r, m:integer;
begin
      ns:=s[0]; na:=a[0];

     if (na>ns) then m:=na else m:=ns;
     r:=0;
     for i:=1 to m do
     begin
          s[i]:=s[i]+a[i]+r;
          r:=s[i] div 26;
          s[i]:=s[i] mod 26;
          a[i]:=0;
     end;
     ns:=m;
```

```
     while (r>0) do
     begin
          ns:=ns+1;
          s[ns]:=r mod 26;
          r:=r div 26;
     end;
     s[0]:=ns;
end;

procedure Scrie;
var i:integer;
begin
     for i:=ns downto 1 do write(g, chr(ord('a')+s[i]));
     writeln(g);
end;

begin
     assign(f,'ilis.in'); reset(f);
     assign(g,'ilis.out'); rewrite(g);
     readln(f, n);
     Citeste;
     for i:=0 to a[0] do begin s[i]:=a[i]; a[i]:=0;end;
     for i:=2 to n do
     begin
          Citeste;
          Suma;
     end;
     Scrie;
     close(f);
     close(g);
end.
```

Clasa a VIII-a

Serial – Filimon Marta **100 puncte**

Ruxandra doreşte ca în următoarea lună să revadă la întâmplare `m` dintre cele `n` episoade ale serialului ei preferat. Episoadele sunt numerotate de la `1` la `n`, în ordinea în care au fost filmate.
Ruxandrei îi plac în mod egal toate episoadele serialului, iar ea nu doreşte să le vadă în ordine crescătoare a apariţiei acestora. Spre exemplu dacă `n=5` si `m=3` nu va viziona episoadele în urmatoarele variante: `(1,2,3) (1,2,4) (1,2,5) (1,3,4) (1,3,5) (1,4,5) (2,3,4) (2,3,5) (2,4,5) (3,4,5)`.

Cerinţă
Să se determine în câte moduri se poate uita Ruxandra la cele `m` episoade.

Date de intrare
Pe prima linie a fişierului serial.in se vor afla `m` şi `n` cu semnificaţiile din enunţ.

Date de ieşire
În fişierul de ieşire se va afla un singur număr reprezentând numărul de posibilităţi modulo `39971`.

Restric ţii şi precizări
$2 \le m \le n \le 100$
Pentru `20%` din punctaj $n \le 10$.

serial.in	serial.out	Explicaţii
5 2	10	{2, 1} {3,1} {4,1} {5,1} {3, 2} {4, 2} {5, 2} {4, 3} {5, 3} {5, 4}
5 3	50	

Timp maxim de execuţie/test: 1 secundă.

Soluţie – Filimon Marta

/* Pentru 20 de puncte: Se vor genera toate posibilităţile şi ale numara pe cele care îndeplinesc condiţia

Pentru 100 de puncte: Vom privi cele n seriale ca fiind o mulţime cu n elemente.

1) Numărul total de submulţimi cu m elemente, ţinând cont de ordinea elementelor din submulţime, dintr-o mulţime cu n este A_n^m(aranjamente de n luate câte m).

2) Dacă nu ţinem cont de ordinea elementelor dintr-o submuţime, înseamnă ca le putem ordona cum dorim. Avantajos pentru noi ar fi să ordonăm crescator elementele din submulţime. Numărul de submulţimi neordonate cu m numere natural reprezintă C_n^m (combinări de n luate câte m).

1) + 2) => Numărul de modalităţi in care se poate uita Ruxandra la episoadele serialului său favorit este: A_n^m- C_n^m

*/

```
program p1;
var f,g:text;
    a:array[1..100, 0..100] of longint;
    n,m,arang,comb:longint;

procedure Numar_aranjamente;
var i:integer;
begin
     arang:=1;
     for i:=n-m+1 to n do arang:=(arang*i) mod 93089;
end;

procedure Numar_combinari;
var i,j:integer;
begin
     for i:=1 to n do a[i][0]:=1;
     a[1][1]:=1;
     for i:=2 to n do
        for j:=1 to i do a[i][j]:=(a[i-1][j-1]+a[i-1][j]) mod 93089;
     comb:=a[n][m];
end;

procedure Scrie;
var rez:longint;
begin
     rez:=(arang-comb+93089) mod 93089;
     writeln(g,rez);
end;

begin
     assign(f,'serial.in');reset(f);
     assign(g,'serial.out');rewrite(g);

     read(f, n, m);

     Numar_aranjamente;

     Numar_combinari;

     Scrie;

     close(f);
     close(g);
end.
```

Clasa a VIII-a

Electro – Filimon Marta **100 puncte**

Pe o tablă de dimensiune 150 000 * 150 000 ce simulează un sistem de coordonate XOY cu originea în partea de stânga jos a tablei, sunt aşezate nrb becuri şi nrg generatoare. Generatoarele măresc cu t intensităţile tuturor becurilor din raza lor de acţiune.
Fiecare generator îşi exercită acţiunea asupra tuturor becurilor aflate în raza lui de acţiune.

Cerinţă

Să se determine numărul de becuri ce au intensitatea maximă precum şi coordonatele lor.

Date de intrare

Pe prima linie a fişierului `electro.in` se găsesc `2` numere naturale `nrb` si `nrg`. Pe următoarele `nrb` linii sunt `2` numere ce reprezintă coordonatele becului. Pe următoarele `nrg` linii se află `4` numere `x`, `y`, `p` şi `t` reprezentând poziţia generatorului, raza pe care acţionează, respectiv intensitatea pe care o dă becurilor.

Date de ieşire

În fişierul `electro.out` se vor afla `2` numere reprezentând intensitatea maximă, respectiv numărul de becuri cu intensitate maximă.

Restricţii şi precizări

- $3 \le nrg, nrb \le 500$
- $1 \le t, p \le 100$
- **Iniţial becurile au intensitatea 0.**

electro.in	electro.out	Explicaţie
7 3 3 4 5 6 11 3 12 9 8 5 5 3 11 7 4 5 2 2 3 3 3 1 10 5 3 3	3 4	Sunt 7 becuri şi 3 generatoare. În urma acţiunii generatoarelor asupra becurilor rezultă 4 becuri care au intensitatea maximă (1,3,5,7). Valoarea maximă a intensităţii finale pentru becurile date este 3.

Timp maxim de execuţie/test: 1 secundă.

Soluţie – Filimon Marta

```
program p1;
type bec=record x,y,o:longint end;
     generator=record x,y:longint; p,o:shortint; end;
var f,g:text;
    b:array[1..1000] of bec;
    a:array[1..1000] of generator;
    nrg,nrb,nr,max:longint;

procedure RRead;
var i:integer;
begin
     read(f,nrb,nrg);
     for i:=1 to nrb do read(f, b[i].x, b[i].y);
     for i:=1 to nrg do read(f, a[i].x, a[i].y, a[i].p, a[i].o);
end;

procedure Ver(i:integer);
var j:integer;
    d, t1, t2:real;
```

```
begin
     for j:=1 to nrb do
     begin
          t1:=(a[i].x-b[j].x);
          t1:=t1*t1;
          t2:=(a[i].y-b[j].y);
          t2:=t2*t2;
          d:=sqrt(t1+t2);;
          if (d<=a[i].p) then b[j].o:=b[j].o+a[i].o;
     end;
end;

procedure Solve;
var i,j:integer;
begin
     for i:=1 to nrg do {pentru fiecare generator in parte voi afla becurile
din raza sa de actiune}
         Ver(i);
     max:=-1; nr:=0;
     for i:=1 to nrb do
         if (b[i].o>max) then begin max:=b[i].o; nr:=1; end
         else if (b[i].o=max) then nr:=nr+1;
end;

procedure WWrite;
var i:integer;
begin
     writeln(g,max,' ',nr);
end;

begin
     assign(f,'electro.in');reset(f);
     assign(g,'electro.out');rewrite(g);

     RRead;

     Solve;

     WWrite;

     close(f);
     close(g);
end.
```

Problemele de concurs 2010

Clasa a V-a

Margele - Mircea Pricop 100 puncte

In fiecare zi, la fabrica de lantisoare din Eltnen sunt premiate cele mai interesante lantisoare. Un lantisor e format din margele de 2 tipuri : rosii si albastre. In functie de configuratia margelelor, fiecare lantisor primeste puncte astfel:

- Daca sunt de doua ori mai multe margele rosii ca albastre, 16 puncte.
- Daca e de forma: X albastre, apoi X rosii, apoi X albastre, din nou, X puncte.
- Daca incepe si se termina cu aceeasi culoare, 4 puncte
- Daca se termina cu doua albastre, 2 puncte
- Daca are doar culori alternante (e de forma "RARA..."), 1 punct.
- Daca are doar margele de o singura culoare, -10 puncte

Cerinta

Avand dat un lantisor, vi se cere sa spuneti cate puncte ar lua la premiere.

Date de intrare

Pe prima linie a fisierului de intrare margele.in se va afla un numar natural N, care reprezinta numarul de margele. Pe fiecare din urmatoarele N linii se va afla fie caracterul "R", fie "A", reprezentand o margea rosie, respectiv albastra.

Date de iesire

Pe prima linie a fisierului de iesire margele.out se va afla scorul adunat de lantisorul dat.

Restrictii si precizari

- 2≤ N ≤ 100000

margele.in	margele.out	explicatii
6 A A R R A A	8	(2 – 2 albastre, 2 rosii, 2 albastre 4 – incepe si se termina cu A 2 – se termina cu 2 A)

Timp maxim de executie/test: 1 secunda

Solutie - Marta Filimon

```
program p1;
var f,g:text;
    n,i,j,ok,lg,x,nra,nrr,p:longint;
    c,pc,first:char;
begin
     assign(f,'margele.in');reset(f);
     assign(g,'margele.out');rewrite(g);
     readln(f,n);
     readln(f,pc);
     i:=1;first:=pc;
     repeat
          i:=i+1;
          readln(f,c);
     until (pc<>c)or(i=n);
     x:=i-1;{x=nr. de margele de aceeasi culoare de la incepul secventei}

     if i=n then
     begin
       if c='A' then nra:=i else nrr:=i;
       ok:=0;{ok=1 daca este alternat si 0 in caz contrar}
     end
     else
     begin
         if pc='A' then begin nra:=i-1;nrr:=1;end
```

```
        else begin nrr:=i-1;nra:=1;end;

        ok:=1;
        j:=i;
        pc:=c;
        lg:=1;
        for i:=j+1 to n do
        begin
             readln(f,c);
             if c=pc then lg:=lg+1
                else
                    begin
                         if lg<>x then ok:=0;
                         lg:=1;
                    end;
             if c='A' then nra:=nra+1
                      else nrr:=nrr+1;
             if i<>n then pc:=c;
        end;
        if lg<>x then ok:=0;
        p:=0;
    end;
    {se determina punctajul final}
    if (pc=c)and(c='A') then p:=2;
    if c=first then p:=p+4;
    if nrr=nra*2 then p:=p+16;
    if (nrr=0)or(nra=0) then p:=p-10;
    if (x>1)and(first='R') then ok:=0;
    if (x=1)and(ok=1) then ok:=2;
    p:=p+x*ok;
    writeln(g,p);
    close(f);
    close(g);
end.
```

Clasele a V-a şi a VI-a

Marmota - Robert Petrov **100 puncte**

O marmota are o gramada mare de alune de greutati diferite si trebuie sa o imparta pentru doua ciocolate. Ea se gandeste sa puna in cele doua ciocolate o cantitate cat mai apropiata de alune. Marmota gandeste astfel : ea pune alune cat mai mari posibile in prima ciocolata astfel incat sa nu depaseasca greutatea celei de-a doua ciocolate. De asemenea, daca la pasul curent aceasta ar adauga o aluna astfel incat sa depaseasca greutatea celei de-a doua ciocolate din greseala, ea se va opri din adaugat alune.

Cerinta

Gandind precum marmota, aflati care este diferenta minima dintre greutatile celor doua ciocolate (intotdeauna≥0) pe care o poate afla marmota.

Date de intrare

Fisierul `marmota.in` contine pe primul rand numarul total de alune `n` si pe al doilea rand, masele alunelor, separate prin spatiu, sortate descrescator.

Date de iesire

Fisierul `marmota.out` trebuie sa contina un numar reprezentand diferenta minima dintre cele doua gramezi.

Restrictii si precizari

- `2≤n≤ 10000`
- Alunele au gramajele strict pozitive, dar mai mici decat `30000`

marmota.in	marmota.out	explicatii
5 8 4 3 2 1	0	Marmota pune in prima ciocolata alune cat mai grele : 8. In acest moment, daca ar gresi si ar pune aluna de greutate 4, diferenta dintre cele doua ar fi 12-6=6 si marmota s-ar opri. Daca adauga aluna de greutate 3, am avea 11-7=4. Daca ar gresi adaugand aluna de greutate 2, am avea 10-8=2. Daca ar adauga aluna de greutate 1, nu ar gresi si am avea 9-9=0. In schimb, s-ar opri deoarece nu ar mai avea alune de adaugat.
13 26 25 24 23 22 22 15 15 14 13 13 12 10	6	Marmota pune in prima ciocolata alune cat mai grele : 26 25 24 23. In momentul acesta, daca ar pune o aluna de greutate 22, ea ar depasi greutatea celei de-a doua ciocolati cu 6 si s-ar opri, diferenta dintre ele fiind 6. Orice alta greseala ar duce spre o diferenta mai mare. Daca ar adauga in mod corect alune 26 25 24 23 15, ar ajunge la 121-113=8, care este mai mare decat 6.

Timp maxim de executie/test: 1 secunda

Solutie –Marta Filimon

```
program p1;
var f,g:text;
    a:array[1..10000] of integer;
    i,j,n,s,c1,c2,max:longint;
begin
     assign(f,'marmota.in');reset(f);
     assign(g,'marmota.out');rewrite(g);
     read(f,n);
     s:=0;
     for i:=1 to n do
     begin
          read(f,a[i]);
          s:=s+a[i];
     end;
     for i:=1 to n do
        if c1+a[i]<=s div 2 then c1:=c1+a[i]
        else break;
     c2:=s-c1;
     max:=abs(c1-c2);
     j:=i;
     for i:=j to n do
     begin
          c1:=c1+a[i];
          c2:=s-c1;
          if abs(c2-c1)<max then max:=abs(c2-c1);
          c1:=c1-a[i];
     end;
     writeln(g,max);
     close(f);
     close(g);
end.
```

Solutia autorului

```
#include <stdio.h>
#define nmax 10000
```

```
FILE *f;
int a[nmax];

int main()
{
	long i,n,s=0,p1,p2,j,x,difmin;
	f=fopen("marmota.in","rt");
	fscanf(f,"%ld\n",&n);
	for (i=1;i<=n;i++)
	{
		fscanf(f,"%d",&a[i]);
		s+=a[i];
	}
	fclose(f);
//luam ca p1>=p2 ... deocamdata
	p1=s;
	p2=0;
	x=s;
	j=0;difmin=x;
	for (i=1;i<=n&&(p2<=s/2);i++)
	{
		if (p2+a[i]<=s/2)//daca adunam, sa nu depaseasca juma'
		{
			p2+=a[i];
			p1-=a[i];
			j=0;//daca s-o inserat ceva, nu mai avem nevoie de adaosul
precedent
			x=s;
		}
		else
//daca se intampla ca p1-p2=diferenta curenta > p2-p1+2*a[i]=diferenta noua
daca transfer a[i]
//identic cu 2*p1-2*p2>2*a[i], adica p1-p2>a[i]
		if (p1-p2>a[i])
		{
			if (p2-p1+2*a[i]<x)
			{
				x=p2-p1+2*a[i];
				j=i;
				if (difmin>x)
					difmin=x;
			}
			//depaseste jumatatea;
		}
	}
	if (j)
	{
		p1-=a[j];
		p2+=a[j];
	}
	p1-=p2;
	if (p1<0)
		p1*=-1;
	if (p1>difmin)
		p1=difmin;
	f=fopen("marmota.out","wt");
	fprintf(f,"%ld",p1);
	fclose(f);
	return 0;
}
```

Clasa a VI-a

Magazine -Bogdan Gâza **100 puncte**

George s-a trezit de dimineata pregatit sa cumpere cadouri pentru `1` Iunie. Acesta, se urca in masina si merge pe strada de unde cumpara de obicei cadouri pentru copii. El parcheaza pe strada, undeva intre primul si ultimul magazin si pentru a lua cadouri cat mai frumoase el decide ce trebuie sa viziteze toate magazinele si apoi sa se intoarca la masina. Pentru ca George este un om ocupat si se grabeste el viziteaza fiecare magazin o singura data. Strada pe care George cauta cadouri este de fapt o linie iar fiecare magazine este identificat printr-un numar natural ce reprezinta pozitia magazinului pe strada.

Cerinta

Ajutati-l pe George realizand un program care ca citi care sa determine distanta minima pe care trebuie o parcurge de la masina, pe la toate magazinele de pe strada si apoi inapoi la masina.

Date de intrare

Fisierul de intrare `magazine.in` contine pe prima linia un numar `N` reprezentand numarul de magazine de pe strada. `A` doua linie contine `N` reprezentand pozitiile magazinelor pe strada. Aceste numere nu sunt neaparat in ordine crescatoare.

Date de iesire

Fisierul de iesire `magazine.out` contine pe prima linie valoare cautata – distant minima parcursa de George.

Restrictii si precizari

- `1<N<65000`
- Pozitia fiecarui magazine pe strada este cuprinsa intre `0` si `65000`.

magazine.in	**magazine.out**
2 2 4	4
4 3 1 5 12	22
3 7 1 3	12

Timp maxim de executie/test: 1 secunda

Solutie – Roxana Luca

```
program magazine;
var f,g:text;
    max,min,a:longint;
    s:longint;
    i,n:word;
begin
     assign(f,'magazine.in');
     assign(g,'magazine.out');
     reset(f);
     rewrite(g);
     read(f,n);
     s:=0; min:=65000; max:=0;
     for i:=1 to n do begin
         read(f,a);
         if min>a then min:=a;
         if max<a then max:=a;
     end;
     s:=(max-min)*2;
     write(g,s);
     close(f);
     close(g);
end.
```

Clasa a VII-a

Galaxy - Vlad Stoian — 100 puncte

V-o prezint pe Natasha, cea mai temuta vanatoare de recompense din galaxia Katana. Aceasta galaxie are forma dreptunghiulara de dimensiuni `m` si `n`, iar in fiecare punct de coordonate intregi $A_{i,j}$ $(0 < i \le m,\ 0 < j \le n)$ se afla o planeta, care ascunde un fugitiv. Natasha isi face planul, care este in felul urmator: ea vrea sa traverseze galaxia pe cele `2` diagonale, si sa recupereze fugitivii din toate planetele care ii ajung in cale, pentru a-i preda si a primi recompensa pusa de pe capul lor.

Cerinta

Stiind valoarea recompensei pusa pe capul fiecarui fugitiv din galaxie, dar si dimensiunile galaxiei, Natasha va roaga sa ii calculati cati bani va aduna, daca toate merg conform planului.

Date de intrare

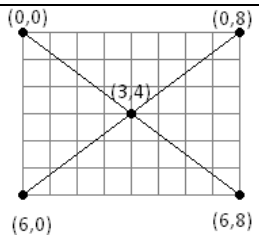
Fisierul de intrare `galaxy.in` contine pe prima linie numarele `n` si `m` reprezentand dimensiunile galaxiei, iar pe urmatoarele `n` linii, `m` valori pe fiecare, reprezentand recompensa pusa pe capul fugitivilor.

Date de iesire

Fisierul de iesire `galaxy.out` trebuie sa contina pe prima linie valoarea cautata.

Restrictii si precizari

- $1 \le n,m \le 100$
- $0 \le A_{i,j} \le 10.000$
- Natasha porneste din punctul `(0,0)`
- Nu se gaseste nici o planeta in punctele de coordonate `(0,x)` si `(x,0)`

galaxy.in	galaxy.out	
6 8 8 3 4 2 5 6 3 4 9 8 5 6 2 3 4 5 0 9 8 3 2 3 4 1 5 7 3 8 6 5 4 6 1 2 3 0 7 8 4 6 3 4 5 6 3 1 3 2	5	 Pe prima diagonala porneste din `(0,0)`, primul punct pe care il intalneste este `(3,4)`, culege de fugitivul pe care va primi `3` zloti iar apoi ajunge in `(6,8)` si il culege pe urmatorul. Pe a doua diagonala, porneste din `(0,8)`, unde nu este nici o planeta, ajunge in `(3,4)`, unde nu mai este nimeni si apoi in `(6,0)`.

Timp maxim de executie/test: 1 secunda

Solutia 1 – Andrei Netedu

```
var f,g:text;
    n,m,i,j,tr1,tr2,x,x1,x2,x3,y1,y2,y3,bn:longint;
begin
assign(f,'galaxy.in');reset(f);
assign(g,'galaxy.out');rewrite(g);
readln(f,n,m);
for i:=1 to n do
    for j:=1 to m do
        begin
        read(f,x);
        x1:=0;
        y1:=0;
        x2:=j;
        y2:=i;
        x3:=m;
        y3:=n;
        tr1:=(x3+x1)*(y3-y1)+(x2+x3)*(y2-y3)-(x2+x1)*(y2-y1);

        x1:=m;
        y1:=0;
        x2:=j;
        y2:=i;
        x3:=0;
        y3:=n;
        tr2:=(x3+x1)*(y3-y1)+(x2+x3)*(y2-y3)-(x2+x1)*(y2-y1);
```

```
        if (tr1=0) or (tr2=0) then bn:=bn+x;
        end;
writeln(g,bn);
close(g);
end.
```

Solutia 2 – Marta Filimon

```
program p1;
var f,g:text;
    a:array[0..100,0..100] of integer;
    i,n,m,j,s,d,x1,y1,x,y:longint;

function cmmdc(x,y:longint):longint;
var r:longint;
begin
     r:=x mod y;
     while r>0 do
     begin
          x:=y;
          y:=r;
          r:=x mod y;
     end;
     cmmdc:=y;
end;

begin
     assign(f,'galaxy.in');reset(f);
     assign(g,'galaxy.out');rewrite(g);
     read(f,n,m);
     for i:=1 to n do
         for j:=1 to m do read(f,a[i,j]);
     d:=cmmdc(n,m);
     s:=0;
     x:=n div d;
     y:=m div d;
     x1:=n-n div d;
     y1:=m div d;
     while x<=n do
     begin
          s:=s+a[x,y];
          a[x,y]:=0;
          s:=s+a[x1,y1];
          a[x1,y1]:=0;
          x:=x+n div d;
          y:=y+m div d;
          x1:=x1-n div d;
          y1:=y1+m div d;
     end;
     writeln(g,s);
     close(f);
     close(g);
end.
```

Clasele a VII-a şi a VIII-a

Cifru - Alexandru Paicu 100 puncte

Link a reusit sa patrunda in castelul maleficului Ganondorf si incearca sa o elibereze pe Zelda. Pentru asta are nevoie sa descifreze cifrul lacatului pus la usa printesei Zelda. Acesta este alcatuit doar din literele mari ale alfabetului englez (`ABCDEFGHIJKLMNOPQRSTUVWXYZ`).
Pentru a reusi, el a primit ca ajutor de la Midna un sir de litere (mari ale alfabetului englez) care poate fi folosit pentru a obtine cifrul adunand la fiecare din litere un numar, acelasi numar pentru fiecare litera din sir. A aduna un numar la o litera inseamna a inlocui acea litera cu litera care se afla dupa atatea pozitii dupa ea in alfabet (ex: `A+5=F`, `M+1=N`, `F+2=H`). De asemenea, daca se ajunge la sfarsitul alfabetului, numararea pozitiilor continua de la inceputul lui (ex: `Z+2=B`, `Y+4=C`).
In plus el a primit mai multe subsiruri de litere despre care stie sigur ca fac parte din cifrul final.

Cerinta

Stiind sirul dat de Midna si subsirurile care fac parte sigur din cifrul final sa se afiseze toate sirurile care ar putea fi cifrul cautat in ordine lexicografica.

Date de intrare

Pe prima linie a fisierului de intrare `cifru.in` se gaseste sirul furnizat de Midna. Pe a doua linie se gaseste un numar natural `K`. Pe ultimele `K` linii se gasesc subsirurile despre care se stie sigur ca fac parte din cifru, cate unul pe linie.

Date de iesire

In fisierul de iesire `cifru.out` se vor gasi toate sirurile posibile care pot fi cifrul cautat.

Restrictii si precizari

- Fiecare din sirurile din fisierul de intrare are maxim `100` de litere.
- `1≤K≤20`

cifru.in	**cifru.out**	**Explicatii**
ABCD 1 FG	DEFG EFGH FGHI	Explicatii: in exemplul `1` putem sa adunam `ABCD` cu `3`, `4` sau `5` si obtinem singurele `3` siruri in care se regaseste subsirul FG. In exemplu `2` singura posibilitate este sa adunam sirul initial cu `5` pentru a obtine un sir in care sa se regaseasca toate subsirurile date.
BMAIROS 4 GR RFN WT GRFN	GRFNWTX	

Timp maxim de executie/test: 1 secunda

Solutia autorului

```
// cifru.cpp : Defines the entry point for the console application.

#include <stdio.h>
#include <string.h>
FILE *fin,*fout;

char s[102];
char sub[20][102];
int ka;

void AddNumber(char *s,int nr)
{int i;
 i=0;
 while (s[i])
      {s[i]=s[i]+nr;
       if (s[i]>'Z') s[i]-=26;
       i++;
      }
}
int main()
{fin=fopen("cifru.in","r");
 fout=fopen("cifru.out","w");
 fscanf(fin,"%s",s);
 int i,j;
 fscanf(fin,"%d",&ka);
 for (i=0;i<ka;++i) fscanf(fin,"%s",sub[i]);
 fclose(fin);
 AddNumber(s,'Z'-s[0]);
 int sw;
 for (j=0;j<26;j++)
      {AddNumber(s,1);
       sw=1;
       for (i=0;i<ka;i++)
```

```
          if (!strstr(s,sub[i]))
                {sw=0;
                 break;
                }
      if (sw==1)
          {fprintf(fout,"%s\n",s);
          }
     }
 fclose(fout);
 return 0;
}
```

Solutia 2 – Marta Filimon

```
program p1;
var f,g:text;
    a:array[1..26] of string[100];
    b:array[1..20] of string[100];
    s,y:string[100];
    i,n,ok,lg,j,m,max,x:longint;
begin
     assign(f,'cifru.in');reset(f);
     assign(g,'cifru.out');rewrite(g);
     readln(f,s);
     readln(f,n);
     for i:=1 to n do readln(f,b[i]);
     lg:=length(s);
     m:=0;max:=ord('Z');
     for i:=0 to 25 do
     begin
          y:='';
          for j:=1 to lg do
          begin
               x:=ord(s[j])+1;
               if x<=max then y:=y+chr(x)
               else y:=y+'A';
          end;
          ok:=1;
          for j:=1 to n do
          if pos(b[j],y)=0 then begin ok:=0;break;end;
          if ok=1 then begin m:=m+1;a[m]:=y;end;
          s:=y;
     end;
     for i:=1 to m-1 do
     for j:=i+1 to m do
     if a[i]>a[j] then
        begin
             y:=a[i];
             a[i]:=a[j];
             a[j]:=y;
        end;
     for i:=1 to m do
     writeln(g,a[i]);
     close(f);
     close(g);
end.
```

Clasa a VIII-a

Bazine - Bogdan Gâza **100 puncte**

Ionica, geologul de serviciu, vrea sa traseze drumul in care se scurge apa prin vaile muntilor ILACEB din China. El numeste drumul in care apa se scurge – **bazin de drenaj.**
El are la dispozitie o matrice patratica de `NxM` in care fiecare locatie reprezinta inaltimea unui pisc din muntii ILACEB.

Cerinta
El trebuie sa noteze fiecare bazin de drenaj cu litere de la `a` la `z` in urmatorul fel:

- Din fiecare locatie a matricii apa curge doar intr-una dintre cele `4` locatii vecine – cea cu valoarea cea mai mica
- Daca o locatie are toti cei `4` vecini mai mari ca ea atunci acea locatie se numeste **lac** si in acest caz apa nu se scurge nicaieri.
- In cazul in care sunt mai multe locatii cu aceeasi valoare minima, se considera locatiile in urmatoarea ordine: Nord, Vest, Est, Sud.

Fiecare locatie din care apa curge direct sau indirect in acelasi lac face parte din acelasi bazin de drenaj. Fiecare bazin are ca nume o litera mica astfel incat daca se **concateneaza** matricea numelor de sus in jos si de la stanga la dreapta, sa se obtina un sir minim din punct de vedere lexicografic.

Date de intrare
Pe prima linie a fisierului de intrare `bazine.in` se gasesc `2` numere `N` si `M` reprezentand numarul de linii respective numarul de coloane ale matricii in cauza. Pe urmatoarele `N` linii se gasesc cate `M` numere care reprezinta locatiile din matrice – despartite
prin spatii.

Date de iesire
In fisierul `bazine.out` trebuie sa se gaseasca o matrice `NxM` – unde fiecare locatie este marcata conform enuntului.

Restrictii si precizari
- `1<N,M<100`
- `0`<altitudini<`10000`
- vor exista cel mult `26` de bazine

bazine.in	**bazine.out**
3 3 9 6 3 5 9 6 3 5 9	a b b a a b a a a
2 13 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	a b c d e f g h i j k l m n o p q r s t u v w x y z

Timp maxim de executie/test: 1 secunda

Solutia 1 – Marta Filimon

```
program p1;
type index=record i,j:shortint;end;
     vect=array[1..4] of shortint;
const di:vect=(-1,0,0,1);
      dj:vect=(0,-1,1,0);
var f,g:text;
    a:array[0..101,0..101] of integer;
    b,d:array[0..100,1..100] of shortint;
    c:array[1..5000] of index;
    fr:array[1..26] of char;
    i,n,m,j,p,u,cod:integer;
    cc:char;

function directie(i,j:integer):integer;
var k,min,pz:integer;
```

```
begin
     min:=a[i,j];pz:=0;
     for k:=1 to 4 do
         if min>a[i+di[k],j+dj[k]] then
            begin
                 min:=a[i+di[k],j+dj[k]];
                 pz:=k;
            end;
     directie:=pz;
end;

procedure adauga(ii,jj:integer);
begin
     u:=u+1;
     c[u].i:=ii;
     c[u].j:=jj;
     d[ii,jj]:=cod;
end;

procedure cauta(x,y:integer);
var k,ii,jj,i1,j1:integer;
begin
     for k:=1 to 4 do
     begin
          ii:=x-di[k];
          jj:=y-dj[k];
          i1:=ii+di[b[ii,jj]];
          j1:=jj+dj[b[ii,jj]];
          if (i1=x)and(j1=y)and(d[ii,jj]=0)and(a[ii,jj]<10001)
              and(b[ii,jj]>0) then adauga(ii,jj);
     end;
end;

procedure bf(i,j:integer);
begin
     p:=1;u:=1;c[p].i:=i;c[p].j:=j;d[c[p].i,c[p].j]:=cod;
     while p<=u do
     begin
          cauta(c[p].i,c[p].j);
          p:=p+1;
     end;
end;

begin
     assign(f,'bazine.in');reset(f);
     assign(g,'bazine.out');rewrite(g);
     read(f,n,m);
     for i:=1 to n do
     for j:=1 to m do
         read(f,a[i,j]);

     for i:=0 to n+1 do
     begin
          a[i,0]:=10001;
          a[i,m+1]:=10001;
     end;
     for j:=1 to m do
     begin
          a[0,j]:=10001;
          a[n+1,j]:=10001;
     end;
```

```
     for i:=1 to n do
     for j:=1 to m do
         b[i,j]:=directie(i,j);

     for i:=1 to n do
     for j:=1 to m do
     if b[i,j]=0 then begin cod:=cod+1;bf(i,j);end;

     cc:='a';
     for i:=1 to n do
     for j:=1 to m do
     if fr[d[i,j]] in ['a'..'z'] then fr[d[i,j]]:=fr[d[i,j]]
     else
        begin
             fr[d[i,j]]:=cc;
             cc:=chr(ord(cc)+1);
        end;
     for i:=1 to n do
     begin
          write(g,fr[d[i,1]]);
          for j:=2 to m do
          write(g,' ',fr[d[i,j]]);
          writeln(g);
     end;
     close(f);
     close(g);
end.
```

Solutia 2 – solutia autorului

```
/*
      BAZINE - Problema 2 Clasa a8-a
    Moisil++ 2010

      Sursa oficiala - Bogdan Gaza
    Problema inspirata dupa problema "Water Sheds" Google Code Jam 2009
    Qualification Round
 */

#include <stdio.h>
#define FIND_MIN 0
#define FIND_MAX 1

int N,M;

int A[100][100];
char flag[100][100];

int offset[4][2] = { {-1,0},{0,-1},{0,1},{1,0} };

// gaseste celula cu valoare MINIMA/MAXIMA din jurul celului date
// problema necesita doar valoare de minim
void applyOffset(int type,int x,int y, int *whereX, int *whereY){
      int i;
      int bestX = x, bestY = y;

      for(i=0;i<4;i++){
            if(x+offset[i][0] >= 0 && x+offset[i][0] < N &&
               y+offset[i][1] >= 0 && y+offset[i][1] < N){
                  if(type == FIND_MIN){
```

```
                    if(A[bestX][bestY] > A[x+offset[i][0]][y+offset[i][1]]){

                         bestX = x+offset[i][0];
                         bestY = y+offset[i][1];
                    }
               }else{
                    if(A[bestX][bestY] < A[x+offset[i][0]][offset[i][1]]){
                         bestX = x+offset[i][0];
                         bestY = y+offset[i][1];
                    }
               }

          }
     }
     *whereX = bestX;
     *whereY = bestY;
}

void markPath(int x,int y,char letter);

/* Cauta vecinii celulei curente ce au ca minim vecin celula curanta */
void applyMark(int x,int y,char letter){
     int i;

     int bestX,bestY;

     for(i=0;i<4;i++){
          if(x+offset[i][0] >= 0 && x+offset[i][0] < N &&
             y+offset[i][1] >= 0 && y+offset[i][1] < N){


     applyOffset(FIND_MIN,x+offset[i][0],y+offset[i][1],&bestX,&bestY);
               if(x==bestX && y==bestY){
                    markPath(x+offset[i][0],y+offset[i][1],letter);
               }

          }
     }
}
/* Aceasta functie daca este aplicata unei celule ce nu a fost marcata deja
 marcheaza recursiv toate celulele care fac parte din acelasi bazin de drenaj.
 Pentru locatia curenta - in primul rand vede daca apa se scurge intr-o alta
locatie.
 Apoi vede daca pentru locatia curenta daca apa mai curge in ea din alte
locatii.
 Aplicatia acest algoritm pentru fiecare celula nemarcata gasita.
 */
void markPath(int x,int y,char letter){
     int bestX,bestY;
     if(flag[x][y] == 0){
          flag[x][y] = letter;
          applyOffset(FIND_MIN,x,y,&bestX,&bestY);
          applyMark(x,y,letter);
          markPath(bestX,bestY,letter);
     }

}


int main () {
     int i,j;
```

```
    freopen("bazine.in","r",stdin);
    freopen("bazine.out","w",stdout);

    scanf("%d %d",&N,&M);

    for(i=0;i<N;i++){
          for(j=0;j<N;j++){
                scanf("%d",&A[i][j]);
          }
    }

    char startLetter = 'a';
    for(i=0;i<N;i++){
          for(j=0;j<N;j++){
                if(flag[i][j] == 0){
                      markPath(i,j,startLetter);
                      startLetter++;

                }
          }
    }
    for(i=0;i<N;i++){
          for(j=0;j<M;j++){
                printf("%c ",flag[i][j]);
          }
          printf("\n");
    }
   return 0;
}
```

Problemele de concurs
Clasa a V-a

Nano – Vlad Stoian **100 puncte**

Nano a descoperit ca deţine o cutie magică care are 3 butoane de comandă: E, U şi M. După mai multe cercetări, şi-a dat seama cum funcţionează. Dacă bagă o singură bilă în cutie şi apasă butonul M, după un timp în cutie vor fi P bile (el numeşte acest proces Multiplicare). Dacă bagă P bile în cutie şi apasă pe U, după un timp va fi una singură (Unire). Dacă în schimb apasă pe butonul E, cutia distruge bilele introduse (acest proces numindu-se Eliminare). Nano execută o serie de procese şi işi notează pe o foaie exact ce a facut. De exemplu: dacă a executat o multiplicare a unui element in P elemente de mai multe ori (X ori) el va scrie: "M X P" (X şi P fiind numere), dacă a executat o unire va scrie "U X P", iar dacă a executat o eliminare, "E P" (a eliminat P elemente).

Cerinţă
Nano ştie că avea N bile la inceput şi ca a executat K operaţii pe ele, precum şi ce operaţii. Ajutaţi-l să afle cu câte bile a ramas după cele K operaţii.

Date de intrare
Fişierul de intrare nano.in conţine pe prima linie numerele N şi K, N fiind numărul de bile, iar K fiind numărul de operaţii. Pe urmatoarele K linii se afla cate o litera (M, E sau U, iniţiala fiecarei operaţii), urmată in funcţie de operatie, de una sau doua cifre, reprezentand P, sau X şi P, cu semnificaţiile din enunţ.

Date de ieşire
Pe prima linie a fişierului de iesire nano.out se va afişa numărul cerut de Nano.

Restrictii şi precizări
- 1 ≤ N ≤ 22.000 1 ≤ K ≤ 44.000
- Operaţiile vor putea fi întotdeauna executate.

nano.in	nano.out	Explicatii
30 4 M 2 5 E 20 U 2 6 E 3	5	In prima fază se execută operaţia **M 2 5.** Se iau 2 bile din cele 30 şi fiecare din ele va fi multiplicată în 5 (deci 28 + 10 = 38). După executarea eliminării mai ramân 38 – 20 = 18 bile. Apoi se execută unirea **U 2 6** ceea ce înseamnă că se introduc 6 bile în cutie, acestea transformându-se în 1, acest proces repetându-se de 2 ori (6 + 2 = 8). După ce se efectuează şi ultima eliminare, ramân 5 bile.

Timp de executie/test: 1 sec.

Solutie
Suma finala va fi retinuta intr-o variabila S, iar problema se rezuma la efectuarea a 3 operatii : la o operatie de Multiplicare se va aduna X*(P-1), la operatia de eliminare se va scadea P, iar la operatia de Unire se va scada X*(P-1).

```
#include <fstream.h>
long n,m,p,x;
char c;

int main()
{long i;

      ifstream f("nano.in");
      ofstream g("nano.out");

      f>>n>>m;

      for (i=1;i<=m;i++)
      {
            f>>c;
            if (c == 'M')
            {
                  f>>x>>p;
                  n+= x * (p-1);
            }
            else if (c == 'E')
```

```
            {
                  f>>x;
                  n-=x;
            }
            else
            {
                  f>>x>>p;
                  n-= x * (p-1);
            }
      }
g<<n<<"\n";

return 0;
}
```

Clasele V-VI

Zoo – Mircea Pricop 100 puncte

Odata cu deschiderea noii gradini zoologice din Moscova, a fost nevoie de cineva care sa puna focile in custile corespunzatoare. Cel mai potrivit pentru aceasta datorie este Iosif, un tanar cu abilitati remarcabile de lucru cu foci.
Gradina dispune de `N` custi, care la inceput sunt goale. Iosif primeste instructiuni in urmatorul mod : I se dau doua numere naturale `X` si `Y`, iar el trebuie sa puna, incepand cu cusca cu numarul `X` si pana la cea cu numarul `Y` (inclusiv) , cate o foca in fiecare cusca.
Totusi, abilitatile lui de lucru cu animalele nu i-au permis sa se dezvolte si in domeniul stiintelor exacte, de aceea Iosif are nevoie de ajutorul tau pentru a realiza un raport in care sa spuna exact cate foci sunt in fiecare cusca dupa executarea tuturor instructiunilor. Pentru acest favor el iti ofera `100` de puncte*.

Date de intrare

Pe prima linie a fisierului `zoo.in` se va afla numarele naturale `N` si `K`. Pe urmatoarele `K` linii se va afla cate o pereche de numere naturale `X` si `Y`, cu semnificatia din enunt.

Date de iesire

Pe prima linie a fisierului `zoo.out` se vor afla `N` numere naturale, reprezentand numarul de foci din fiecare cusca dupa introducerea tuturor focilor.

Restrictii si precizari

$1 \le N \le 29\ 000$; $1 \le K \le 29\ 000$; $1 \le X \le Y \le N$

zoo.in	zoo.out	Explicatii
5 4 2 4 1 5 5 5 4 5	1 2 2 3 3	Dupa prima instructiune: 0 1 1 1 0 Dupa a doua: 1 2 2 2 1 Dupa a treia: 1 2 2 2 2 Dupa a patra: 1 2 2 3 3

Timp de executie/test: 1 sec.

* Punctele vor fi acordate in viitor, in urma unei evaluari a codului sursa prezentat de o comise de specialitate special pregatita.

Solutie

Solutia naiva ar fi sa retinem numerele intr-un vector iar la fiecare pas sa parcurgem elementele intre X si Y si sa le incrementam:

```
#include <fstream.h>
int s,n,k,i,x,y,a[29002],j;
int main()
{
ifstream fin("zoo.in");
ofstream fout("zoo.out");
fin>>n>>k;
for (i=1;i<=k;i++)
    { fin>>x>>y;
```

```
      for (j=x;j<=y;j++)
      a[j]++;
    }
for (i=1;i<=n;i++)
    {
      fout<<a[i]<<" ";
    }
fout<<endl;
return 0;
}
```

O solutie mai eficienta ar fi sa folosim un vector de incrementare. Parcurgem vectorul retinand o variabila s, care va reprezenta valoarea curenta. Cand citim un interval, incrementam valoarea de pe pozitia X si decrementam cea de pe pozitia Y. Cand parcurgem vectorul, adunam la fiecare pas in s valoarea de pe pozitia curenta din vectorul de incrementare. Astfel, la inceputul unui interval, s va fi incrementat, toate valorile din interior fiind "marite", iar la sfarsit se va modifica la loc.

```
#include <fstream.h>
int s,n,k,i,x,y,a[29002];
int main()
{
ifstream fin("zoo.in");
ofstream fout("zoo.out");
fin>>n>>k;
for (i=1;i<=k;i++)
    { fin>>x>>y;
      a[x]++;
      a[y+1]--;
    }
s=0;
for (i=1;i<=n;i++)
    { s+=a[i];
      fout<<s<<" ";
    }
fout<<endl;
return 0; }
```

Clasele VI-VII-VIII

Math – Robert Petrov 100 puncte

Leshrac are o obsesie pentru numerele prime. El doreste sa obtina un numar prim mai mic sau egal cu `60000` din orice numar, folosind de cate ori este nevoie, doua operatii: daca numarul este par, el il imparte la `2`, daca este impar il inmulteste cu `3` si aduna `1`. Ajutati-l pe Leshrac sa afle numarul minim de operatii pe care trebuie sa le execute pe `N` numere, pentru a le face prime.

Cerinta

Dandu-se `N` numere, se cere sa se afle, pentru fiecare din ele, numarul minim de operatii necesare pentru a obtine un numar prim.

Date de intrar

Pe prima linie a fisierului de intrare `math.in` se afla numarul natural `N`, iar pe a doua linie `N` numere naturale separate prin cate un spatiu.

Date de iesire

Pe prima si singura linie a fisierului de iesire `math.out` se vor afla `N` numere separate prin cate un spatiu reprezentand numarul cerut pentru fiecare din cele `N` numere, in ordinea lor din fisierul de intrare.

Restrictii si precizari

- `1 ≤ N ≤ 700.000`
- Cele `N` numere sunt mai mici de `60.000` si strict mai mari decat `0`.

math.in	math.out	explicatie
4 2 102 27 33	0 7 2 6	2 este numar prim. 102->51->154->77->232->116->58->29 este numar prim 27->82->41 este numar prim. 33->100->50->25->76->38->19 este numar prim.

Timp de executie/test: 1 sec.

Soluţie

Avem functia f(x) = x/2, daca x este par, si x*3+ 1, daca x este impar.
O solutie ar fi sa luam fiecare numar si sa aplicam functia f pe el, pana ce obtinem un numar prim. Putem verifica daca un numar este prim prin mai multe metode, metode ce vor influenta punctajul final. Cea mai naiva metoda este de a parcurge numerele mai mici ca X/2 pana gasim un divizor. O astfel de abordare ia 20 de puncte. Putem parcurge numerele si pana la radical din X, solutie ce ia 40 de puncte. O solutie de 70 de puncte consta in preprocesarea tuturor numerelor prime pana la 60000, pentru ca apoi verificarea primalitatii unui numar sa o efectuam in timp constant. Astfel vom cauta divizori numai in vectorul de numere prime. Pentru 100 de puncte, trebuie sa optimizam solutia de 70, respectiv modul in care determinam daca un numar este prim. Vom afla in timp constant memorand intr-un vector primalitatea fiecarui numar. Vom construi acest vector folosind ciurul lui Erathostenes.

```
#include <stdio.h>
#define NM 60001
int n;
char v[NM];
int main()
{freopen("math.in","r",stdin);
 freopen("math.out","w",stdout);
 int i,j,x;
 v[i]=1;
 for (i=2;i<NM;i++)
      if (!v[i])
            {for (j=i+i;j<NM;j+=i)
                  {v[j]=1;
                  }
            }
 scanf("%d",&n);
 int nr;
 for (i=1;i<=n;i++)
      {scanf("%d",&x);
       nr=0;
       while (x>=NM||v[x]==1)
             {nr++;
              if (x%2==0)
                    x/=2;
                 else
                    x=x*3+1;
             }
       printf("%d ",nr);
      }
 return 0;
}
```

Clasele VII-VII

Omulet – Alexandru Paicu 100 puncte

Un omulet este prins intr-un joc pe calculator. In acel joc el trebuie sa ajunga la una dintre iesiri, pentru a avansa la nivelul urmator. Miscarile pe care le poate face sunt la stanga si la dreapta, deplasandu-se in casute libere din stanga sau dreapta lui. In plus, daca nu sta pe o platforma (daca se afla in aer) atunci el “cade” si nu poate sa se miste nici in stanga, nici in dreapta pana ce nu ajunge pe o suprafata solida.
Pentru a-l ajuta sa ajunga la iesire, creatorul jocului a decis sa amplaseze portale in anumite casute. Portalele sunt formate din doua capete legate intre ele. In cazul in care omuletul ajunge la un capat al unui portal, el poate decide sa se teleporteze prin el, si sa ajunga instant la celalt capat al acelui portal. Desigur, nu este obligat sa se teleporteze prin

el daca nu vrea. Intentia lui este sa treaca prin cat mai putine portale. Astfel, el va da o matrice care reprezinta nivelul la care a ajuns si va roaga sa-l ajutati.

Cerinta:
Dandu-se un nivel al jocului, sa se determine numarul minim de portale prin care trebuie sa treaca omuletul pentru a ajunge la o iesire.

Date de intrare:
Pe prima linie a fisierului **omulet.in** se gasesc N si M, numarul de linii respectiv de coloane ale matricii. Pe urmatoarele N linii se gasesc cate M caractere avand urmatoarea semnificatie:

"T" – pozitia de start a omuletului;

"X" – una dintre iesiri din nivelul curent;

"P" – platforma (pe care omuletul poate sta), el nu poate trece prin casute care sunt ocupate de platforme;

O cifra (intre 0 si 9) – capatul unui portal (fiecare cifra arata din ce portal face parte, astfel spre exemplu cifrele de 1 sunt capetele portalului cu numarul 1);

"S" – spatiu liber.

Date de iesire:
Pe prima linie a fisierului **omulet.out** se gaseste un numar care reprezinta numarul minim de portale prin care trebuie sa treaca omuletul. In cazul in care nu exista o modalitate pentru ca el sa ajunga la o iesire, se va afisa -1.

Restrictii si precizari:
1≤N, M≤25

Orice cifra care va aparea in matrice va aparea exact de 2 ori

Vor exista maxim 10 portale

Ultima linie din matrice va fi alcatuita doar din "P"

Va exista un singur "T"

Omuletul nu are voie sa iasa din matrice.

omulet.in	omulet.out	Explicatii
2 5 STPXX PPPPP	-1	Nu exista nici o posibilitate ca omuletul sa ajunga la nici una dintre iesiri.
4 8 TPSXSX21 1PSPSPPS SPPPPP2S PPPPPPPP	2	Cade o patratica dupa care se teleporteaza prin portalul 1 ajungand la capatul din dreapta-sus. De acolo cade doua patratele, face o miscare spre stanga si se teleporteaza prin portalul 2 ajungand la capatul de deasupra. Mai face o miscare la stanga si ajunge la iesire.

Timp de executie/test: 1 sec.

Soluţie

Se foloseste o varianta modificata a algoritmului lui Lee.

```
#include <stdio.h>
#define NM 26
#define INF 12
int n,m;
char a[NM][NM];
int b[NM][NM];
struct punct{int x,y;
          };
struct portal{punct A,B;
           } port[11];

int st,dr;
punct c[NM*NM*10];//!!!! fara '*10' doar 90 pct

void bf(int,int);

void add(punct,int);
```

```
inline int egale(punct A,punct B)
{if (A.x!=B.x) return 0;
 if (A.y!=B.y) return 0;
 return 1;
}

inline int inport(punct D)
{int i;
 for (i=0;i<=9;i++)
    if (port[i].A.x!=0)
      {if (egale(port[i].A,D)||egale(port[i].B,D))
                 {return i;
                 }
      }
 return -1;
}

int main()
{freopen("omulet.in","r",stdin);
 freopen("omulet.out","w",stdout);
 scanf("%d %d\n",&n,&m);
 int i,j;
 for (i=1;i<=n;i++) scanf("%s\n",a[i]+1);

 for (i=1;i<=n;i++)
       for (j=1;j<=m;j++)
            b[i][j]=INF;

 for (i=1;i<=n;i++)//constructing the portal vector
   for (j=1;j<=m;j++)
            if (a[i][j]>='0'&&a[i][j]<='9')//if it's the first end
                 {if (port[a[i][j]-'0'].A.x==0)
                       {port[a[i][j]-'0'].A.x=i;
                        port[a[i][j]-'0'].A.y=j;
                       }
                      else//if it's the second end
                       {port[a[i][j]-'0'].B.x=i;
                        port[a[i][j]-'0'].B.y=j;
                       }
                 }
 for (i=1;i<=n;i++)
       for (j=1;j<=m;j++)
            if (a[i][j]=='T')
                 {bf(i,j);
                 }

 int min=INF;
 for (i=1;i<=n;i++)
       for (j=1;j<=m;j++)
            if (a[i][j]=='X'&&b[i][j]<min) min=b[i][j];//checking all the exits

 if (min==INF) printf("-1");
       else printf("%d",min);
 return 0;
}

void bf(int xx,int yy)
{punct E,F;
 st=1,dr=1;
 b[xx][yy]=0;
 c[1].x=xx;
```

```
 c[1].y=yy;
 int p;
 while (st<=dr)
       {E=c[st++];
        if (a[E.x][E.y]=='X') continue;//if it's an exit, useless to expand
        p=inport(E);
        if (p!=-1)//portal
              {punct R,S;
               R=port[p].A;
               S=port[p].B;
               if(egale(E,R))add(S,b[E.x][E.y]+1);
               else add(R,b[E.x][E.y]+1);
              }

        if (a[E.x+1][E.y]=='P')//st si dr
              {F=E;
               F.y++;   //dreapta
               if (F.y<=m&&a[F.x][F.y]!='P')
                     {add(F,b[E.x][E.y]);
                     }
               F.y-=2;  //stanga
               if (F.y>=1&&a[F.x][F.y]!='P')
                     {add(F,b[E.x][E.y]);
                     }
              }
           else//cadere
              {F=E;
               F.x++;
               add(F,b[E.x][E.y]);
              }
       }
}
void add(punct A,int v)
{if (b[A.x][A.y]>v)
      {dr++;
       b[A.x][A.y]=v;
       c[dr]=A;
      }
}
```

Problemele de concurs 2008

Clasa a V-a

Lumini – Stoian Vlad **100 puncte**

Papucel este angajatul unei firme de constructii formata din N zone. El lucreaza la un panou de control cu multe butoane ce poate modifica starea becurilor din cele N zone. In pauza dintre schimburi, seful lui ii trimite scris pe o foaie de hartie cate butoane sa actioneze precum si numarul de camere si camerele in care butonul schimba starea luminii.

Cerinta

Fiind dat numarul N, starea initiala a celor N camere, cat si numarul de butoane si informatiile despre fiecare buton, sa se afle numarul de camere care au lumina aprinsa dupa actionarea celor M butoane si indicele acestora.

Date de intrare

Fisierul de intrare lumini.in contine pe prima linie numarul N reprezentand numarul de camere, pe a doua linie starea fiecarei camere (1, daca lumina este aprinsa, 0, daca lumina este stinsa), pe a treia linie numarul M reprezentand cate butoane trebuie apasate, iar pe urmatoarele M linii informatiile despre butoane de forma NR X_1 X_2 .. X_{NR} . NR reprezinta numarul de camere asupra carora actioneaza butonul, iar X_1… X_{NR} indicele camerelor asupra carora actioneaza.

Date de iesire

Fisierul de iesire lumini.out trebuie sa contina pe prima linie un numar natural S reprezentand numarul de camere cu lumina aprinsa, iar pe a doua linie se vor afla indicele camerelor care au lumina aprinsa ordine crescatoare.

Restrictii si precizari

- $3 \le N \le 3.000$; $1<M<N$
- Schimbarea starii de lumina se face astfel: daca intr-o camera este lumina (are valoarea 1), va deveni intuneric (va lua valoarea 0) si invers.

lumini.in	lumini.out
10 1 1 0 1 0 0 0 0 1 1 1 2 4 9	3 1 2 10

Timp executie pe test: 1 secunda

Solutie

```
program lumini;
var x:array[1..3000] of integer;
    n,m,i,j,nr,u:integer;
    f,g:text;
begin
 assign(f,'lumini.in');reset(f);
 assign(g,'lumini.out');rewrite(g);
 readln(f,n);
 for i:=1 to n do
        read(f,x[i]);
 readln(f,m);
 for i:=1 to m do
  begin
  read(f,nr);
  for j:=1 to nr do begin
                    read(f,u);
                    x[u]:=1-x[u];
                    end;
  readln(f);
  end; nr:=0;
  for i:=1 to n do
     nr:=nr+x[i];
  writeln(g,nr);
  for i:=1 to n do
    if x[i]=1 then write(g,i,' ');
  close(f);
  close(g);
end.
```

Clasele V-VI

Alegeri–Petrov Robert **100 puncte**

Pe planeta Marte, fostul conducator a murit si a venit timpul pentru alegerea unui alt lider. Aici populatia este impartita in `N` grupuri cu `a[1], a[2] … a[N]` martieni. Un martian candidat poate deveni lider doar daca este votat de jumatate plus unu din grupe. De asemenea, un candidat este votat de un grup doar daca este votat de jumatate plus unu din totalitatea martienilor din acel grup. Candidatul Base vrea sa castige alegerile, iar pentru acest lucru, el trebuie sa afle care este numarul minim de martieni care trebuie sa-l voteze pentru a putea ajunge noul conducator al planetei Marte.

Cerinta

Ajutati-l pe candidatul Base sa afle numarul minim de martieni ce trebuie sa-l voteze pentru a putea deveni noul lider.

Date de intrare

Fisierul de intrare `alegeri.in` contine pe prima linie numarul natural `N` reprezentand numarul de grupuri. Urmatoarea linie contine `N` numere naturale reprezentand cati alegatori sunt in fiecare dintre grupe.

Date de iesire

Fisierul de iesire `alegeri.out` trebuie sa contina pe prima linie un numar natural `S` reprezentand cati alegatori trebuie sa convinga Base sa-l voteze pentru a castiga alegerile. Pe cea de-a doua linie se vor afisa indicii grupurilor, in ordine crescatoare, din care fac parte cei `S` alegatori.

Restrictii si precizari

- `3 ≤ N ≤ 10.000`
- Pentru orice `i=1..N, 0<a[i]<1000`

alegeri.in	**alegeri.out**
3 5 7 5	6 1 3
4 3 8 9 8	12 1 2 4

Timp executie pe test: 1 secunda

Solutie

```
program alegeri;
var a,b,c:array[1..10000] of integer;
    n,i,j,aux,x:integer;
    aux1:byte;
    s,sv:longint;
    f,g:text;
begin
 assign(f,'alegeri.in');reset(f);
 assign(g,'alegeri.out');rewrite(g);
 readln(f,n);
 for i:=1 to n do
     begin
     read(f,x);
     a[i]:=x div 2+1;
     b[i]:=i;
     end;

 for i:=1 to n-1 do
   for j:=i+1 to n do
     if a[i]>a[j] then begin
                       aux:=a[i];
                       a[i]:=a[j];
                       a[j]:=aux;
                       aux:=b[i];
                       b[i]:=b[j];
                       b[j]:=aux;
                       end;
 s:=0;
```

```
 for i:=1 to n div 2+1 do
    begin
    s:=s+a[i];
    c[b[i]]:=1;
    end;

 writeln(g,s);
 for i:=1 to n do
   if c[i]=1 then begin
                  write(g,i,' ');
                  end;
 close(f);
 close(g);
end.
```

Clasele VI-VIII

Cersetor–Paicu Alexandru **100 puncte**

Intr-un oras utopic exista o piata dreptunghiulara de dimensiuni `N` si `M`. Aceasta piata este impartita in `NxM` sectoare patrate de dimensiuni `1x1`. Un cersetor observa ca in unele patrate se afla domni generosi care i-ar da o suma de bani daca ar ajunge in patratul lor, iar in altele se afla niste vamesi care il obliga sa plateasca o suma de bani pentru a intra in patratul lor. Din cauza ca cersetorul nostru are niste principii mai stranii, el doreste sa porneasca dintr-un patratel al pietei (nu neaparat de pe margine) si sa mearga de fiecare data in patratelul care se afla in Sud-Est (dreapta-jos) fata de cel in care se afla. Deasemenea el poate sa se opreasca cand vrea si sa ramana in acel patrat pana se face seara si pleaca toti vamesii si domnii generosi.
O ultima dorinta a cersetorului ar fi ca suma pe care o strange sa fie maxima.

Cerinta
Fiind date sumele de bani din fiecare patrat al pietii (pozitive cand cersetorul primeste bani si negative cand el trebuie sa dea bani), sa se calculeze suma maxima pe care o poate obtine cersetorul respectandu-si principiile.

Date de intrare
Pe prima linie a fisierului **`cersetor.in`** se gasesc doua numere `N` si `M`. Pe fiecare din urmatoarele `N` linii se gasesc `M` numere reprezentand banii pe care ii primeste cersetorul daca trece prin acel patratel (daca este un numar negativ inseamna ca trebuie sa plateasca bani).

Date de iesire
Pe prima linie a fisierului **`cersetor.out`** se va gasi suma maxim obtinuta.

Restrictii si precizari
`1≤N, M≤100`
Este obligatoriu ca cersetorul sa parcurga cel putin un patrat.
Se considera un drum valid si un drum in care cersetorul se opreste chiar in casuta din care incepe.
Elementele din matrice sunt in intervalul `[-100, 100]`

cersetor.in	**cersetor.out**	**explicatii**
3 5 1 -1 3 -1 0 2 3 2 2 1 1 4 -5 5 -1	7	porneste din (2,3) si se opreste in (3,4) 2+5=7
3 4 1 0 2 3 0 1 2 3 5 2 4 3	6	porneste din (1,1) si se opreste in (3,3) 1+1+4=6

Timp maxim de executie/test: 1 secunda

Solutie

```
program cersetor;
var a:array[1..100,1..100] of integer;
    n,m,i,j,k:integer;
    max,s:longint;
    f,g:text;
```

```
begin
assign(f,'cersetor.in');reset(f);
assign(g,'cersetor.out');rewrite(g);
readln(f,n,m);
for i:=1 to n do
 begin
 for j:=1 to m do
   read(f,a[i,j]);
 readln(f);
 end;
 max:=a[1,1];
 for i:=1 to n do
   for j:=1 to m do
      begin
      s:=0;
      k:=0;
      while (i+k<=n)and(j+k<=m) do
            begin
            s:=s+a[i+k,j+k];
            if s>max then max:=s;
            k:=k+1;
            end;
      end;
 writeln(g,max);
 close(f);
 close(g);
end.
```

Clasele VII-VIII

Portocale–Stoian Vlad 100 puncte

La conventia anuala a mancatorilor de fructe se intalnesc `n` persoane. Organizatorii au la dispozitie `N` cutii de portocale. Dorind ca toate persoanele sa primeasca acelasi numar de portocale, seful conventiei vine cu ideea sa se deschida anumite cutii astfel incat intregul lor continut sa se distribuie in mod egal tuturor persoanelor, deci sa nu ramana portocale in cutiile deschise.

Cerinta

Fiind dat numarul `N` de persoane cat si numarul de portocale din fiecare din cele `n` cutii, sa se afle una dintre variantele posibile in care pot fi alese cutiile ce vor fi deschise, pentru a veni in sprijinul conventiei.

Date de intrare

Fisierul de intrare `porto.in` contine pe prima linie numarul `N` reprezentand numarul de persoane participante la conventie, iar pe a doua linie `N` numere x_1, x_2 ... x_N, reprezentand numarul de portocale din cele `N` cutii.

Date de iesire

Fisierul de iesire `porto.`*`out`* trebuie sa contina pe prima linie un numar natural *NR*, reprezentand numarul de cutii ce se vor desface, iar pe a doua linie, *NR* numere separate prin cate un spatiu, reprezentand numerele de ordine ale cutiilor ce vor fi deschise.

Restrictii si precizari

- $3 \le N \le 3000$
- $1 \le x \le 5000$
- Solutia nu este neaparat unica.

porto.in	porto.out	Explicatii
5 2 7 6 10 4	3 3 4 5	S-au desfacut cutiile persoanelor 3, 4 si 5, s-au obtinut astfel 6+10+4=20 portocale care se pot distribui in mod egal celor 5 persoane (fiecareia revenindu-i 4 portocale).

Timp executie pe test: 1 secunda

Solutie

```
program portocale;
var a,r:array[1..10000] of integer;
    n,i,j,m,ok,k:integer;
    f,g:text;
begin
 assign(f,'porto.in');reset(f);
 assign(g,'porto.out');rewrite(g);
 readln(f,n);
 for i:=1 to n do read(f,a[i]);
 r[1]:=a[1] mod n;
 for i:=2 to n do r[i]:=(r[i-1]+a[i]) mod n;
 ok:=0;
 for i:=1 to n do
   if r[i]=0 then begin
                  ok:=1;
                  m:=i;
                  writeln(g,m);
                  for j:=1 to i do write(g,j,' ');
                  break;
                  end;
 if ok=0 then
    begin
    for i:=1 to n-1 do
     if ok=0 then for j:=i+1 to n do
                    if r[i]=r[j] then
                         begin
                         ok:=1;
                         m:=j-i;
                         writeln(g,m);
                         for k:=i+1 to j do write(g,k,' ');
                         break;
                         end;
    end;
    close(f);
    close(g);
end.
```

Problemele de concurs 2007
Clasa a V-a

Balta magică- Robert Petrov

În vremea antică exista o baltă în care evoluţia speciilor era accelerată. Iniţial, în baltă nu se afla nici o fiinţă. Într-o zi veni un bătrân lângă baltă şi grăi: ”De acum încolo, câte zile voi mai trăi, voi aduce câte un peşte în fiecare zi”. Această baltă avea proprietăţi magice. Zilnic fiecare peşte îşi făcea o clonă, iar apoi peştii existenţi din baltă se reproduceau rezultând tot atâţia peştişori (pui) câţi erau cu două zile în urmă, de fiecare dată când bătrânul arunca un peşte în apă, dar înainte ca acesta să ajungă în baltă. Puii deveneau maturi în dimineaţa zilei următoare.

Cerinţă

Aflaţi câţi peşti sunt în baltă în cea de-a `n`-a zi, după vizita bătrânului, ştiind că el şi-a ţinut promisiunea.

Date de intrare

Se citeşte de la tastatură un număr natural `n`.

Date de ieşire

Se va afişa pe ecran numărul de peşti care se găsesc în baltă în a `n`-a zi, după plecarea bătrânului.

Restricţii şi precizări

`1≤n≤100`

Peştii sunt nemuritori.

Intrare	Ieşire
1	1
2	3
3	8

Timp maxim de execuţie/test: 1 secundă

Solutie

Notăm cu a[i] numărul de pesti care se gasesc în baltă în a i-a zi. Din enunt deducem că a[1]=1 şi `a[i]=2*a[i-1]+a[i-2]+1`.

```
program balta;
var a:array[0..100] of longint;
    n,i:integer;
    f,g:text;
begin
 assign(f,'balta.in'); reset(f);
 assign(g,'balta.out');rewrite(g);
 readln(f,n);
 close(f);
 a[0]:=0;
 a[1]:=1;
 for i:=2 to n do a[i]:=2*a[i-1]+a[i-2]+1;
 writeln(g,a[n]);
 close(g);
end.
```

Templu-Mircea Pricop 100 puncte

La templul Wu-Tang s-au adunat n candidaţi care vor să urmeze cursurile de arte marţiale. Din păcate printre candidaţi s-au amestecat câţiva spioni Shaolin. Maestrul i-a aliniat pe toţi şi a reuşit să îşi dea seama pentru fiecare din ei dacă sunt spioni sau candidaţi buni. Deoarece are o mentalitate paşnică, el va încerca să îşi aleagă studenţii fără a alarma pe nimeni, astfel: va alege cea mai lungă secvenţă de candidaţi oneşti şi îi va lua în clan, apoi dintre cei rămaşi va mai alege încă o dată cea mai lungă secvenţă de candidaţi oneşti şi îi va lua şi pe ei.

Date de intrare

De la tastatură se citeşte `N`, numărul de candidaţi, apoi `N` numere de `0` şi `1`, reprezentând: `0`, dacă candidatul este un spion, respectiv `1` dacă este candidat onest.

Date de ieşire

Pe ecran se va afişa un singur număr, reprezentând numărul de candidaţi oneşti aleşi după regula maestrului.

Restricţii

- 1 ≤ N ≤ 100

6 1 1 0 1 1 1	5
13 0 1 1 1 0 1 1 1 1 0 1 1 0	7

Timp maxim de execuţie/test: 1 secundă

Soluţie

Se folosesc 3 variabile: sc, smax şi smax1, reprezentand lungimea secentei, curente de elemente egale cu 1, lungime secventei maxime de elemente egale cu 1 si a celei de-a doua secvente ca lungime. Când începe o nouă secvenţă de elemente egale cu 1 (adica a[i] este 1), se determină lungimea acesteia, pentru parcurgerea ei folosindu-se indicele j. Când secvenţa s-a terminat, se modifică, dacă este cazul, valorile smax şi/sau smax1.

```
var n,i,x,j,sc,smax,smax1:byte;
a:array[1..100] of integer;
    f,g:text;
begin
 assign(f,'templu.in'); reset(f);
 assign(g,'templu.out');rewrite(g);
 readln(f,n);
 for i:=1 to n do read(f,a[i]);
smax:=0; smax1:=0;
 i:=1;
 while i<=n do
  if a[i]=1 then begin
                 sc:=0;
                 j:=i;
                 while (j<=n)and(a[j]=1) do
                     begin
                     sc:=sc+1;
                     j:=j+1;
                     end;
                if sc>=smax then begin
                                 smax1:=smax;
                                 smax:=sc;
                                 end
                else if sc>smax1 then smax1:=sc;
                i:=j+1;
                end
  else i:=i+1;
 writeln(g,smax+smax1);
close(f);
close(g);
end.
```

Clasa a VI-a

Mere-Stoian Vlad **100 puncte**

Fermierul Petrică are o livadă în formă de pătrat cu latura n, împărţită în n x n zone de latură 1. În fiecare zonă se află o cantitate de mere. A venit timpul recoltei iar fermierul, ştiind ce drum trebuie să urmeze pentru a avea recoltă bogată şi anul viitor, vrea sa ştie câte mere va strânge, ca să işi ia destule coşuri cu el.

Cerinţa

Fiind date cantităţile de mere din fiecare zonă, cât şi mărimea livezii, pentru o zonă de plecare şi un drum dat, aflaţi câte mere va strânge Petrică.

Date de intrare

Pe prima linie din fişierul mere.in se va afla numărul natural n reprezentând latura livezii şi numărul natural m reprezentând numărul de instrucţiuni, pe a doua linie se vor afla numerele S_i şi S_j , reprezentând linia şi coloana punctului de plecare. Pe următoarele n linii se vor afla câte n numere separate prin spaţii, reprezentând numărul de mere din fiecare zonă iar pe ultimele m linii se vor afla instrucţiunile de forma: una din literele N,S,E,V reprezentând direcţia (sus, jos, dreapta, stânga) urmată de un număr reprezentând numărul de paşi în acea direcţie.

Date de ieşire

Pe prima linie din fişierul `mere.out` se va afla o valoare reprezentând numărul de mere culese.

Restricţii si precizări

- $1 \le N, M, S_j, S_j \le 75$
- Numărul de mere din fiecare zonă este între `0` şi `100`
- Instrucţiunile nu îl vor trimite pe Petrică în afara livezii.
- Odată culese merele dintr-o zonă, nu vor mai putea fi culese a doua oara.

Mere.in	**mere.out**	**Explicaţii**
5 4 1 2 5 6 7 3 4 2 9 4 0 1 5 7 9 3 6 10 3 4 1 16 4 5 2 7 21 S 4 V 1 N 3 E 2	55	5 **6** 7 3 4 Petrică va porni din zona a doua **2** **9** **4** 0 1 de pe primul rând (cu valoarea 6) **5** **7** 9 3 6 si va merge: 4 zone în jos **10** **3** 4 1 16 (9+7+3+5), una în stânga (4), 3 în **4** **5** 2 7 21 sus (10+5+2) iar în final, 2 zone la dreapta, dar de data aceasta, în căsuţa (2,2) nu va mai găsi nici un măr (0+4). În total vor fi 55 de mere culese.

Timp maxim de execuţie/test: 1 secundă

Soluţie

În variabila `s` se calculează numărul total de mere. Se simulează drumul pe care Petrică trebuie să îl parcurgă conform instrucţiunilor. Când se ajunge într-o poziţie a traseului, se culeg toate merele găsite acolo şi se lasă în urmă valoarea`0`.

```
program mere;
var a:array[1..75,1..75] of integer;
    n,m,i,j,k,si,sj,nr:integer;
    s:longint;
    dir:char; {directia de deplasare}
    f,g:text;
begin
assign(f,'mere.in');reset(f);
assign(g,'mere.out');rewrite(g);
readln(f,n,m);
readln(f,si,sj);

for i:=1 to n do
  begin
  for j:=1 to n do read(f,a[i,j]);
  readln(f);
  end;
s:=a[si,sj];
a[si,sj]:=0;
for k:=1 to m do
  begin
  readln(f,dir,nr);
  for j:=1 to nr do
    begin
    if dir='N' then si:=si-1
    else if dir='S' then si:=si+1
         else if dir='E' then sj:=sj+1
              else sj:=sj-1;
    s:=s+a[si,sj];
    a[si,sj]:=0;
    end;
  end;
 writeln(g,s);
 close(g);
 close(f);
end.
```

Clasele VII-VIII

Marte-Robert Petrov **100 puncte**

Relaţiile de sânge dintre marţieni sunt foarte confuze. Un marţian poate avea un părinte dar şi zece. Nimeni nu va fi surprins dacă cineva are o sută de copii. În consiliul planetar dacă cineva confundă arborele genealogic el se face de ruşine. În acest consiliu se întalnesc cei mai de seamă marţieni şi deci pentru a nu ofensa pe cineva, primii care vor ţine discursuri vor fi marţienii cei mai bătrâni. Un marţian îşi ştie doar părinţii lui, dar nu şi bunicii.

Cerinţă

Să se afişeze ordinea în care ar trebui să ţină discursul marţienii pentru a nu ofensa pe nimeni.

Date de intrare

Pe prima linie a fişierului de intrare `marte.in` se află numărul natural `n` de marţieni. Pe liniile `i=2..n+1` ale fişierului se vor afla copiii marţianului `i`, fiecare rând terminându-se cu `0`.

Date de ieşire

Pe prima şi singura linie a fişierului de ieşire `marte.out` se va afla ordinea în care vor ţine marţienii discursurile.

Restricţii şi precizări

`N≤100`

Soluţia nu este neapărat unică.

marte.in	marte.out
5 0 4 5 1 0 1 0 5 3 0 3 0	2 4 5 3 1

Timp maxim de execuţie/test: 1 secundă

Soluţie

Codificăm relaţiile tată-fiu într-o matrice pătratică cu `n` linii şi `n` coloane, astfel: `a[i,j]` va fi `1` dacă şi numai dacă marţianul `i` este tatăl marţianului `j`. Determinăm pentru fiecare marţian `j`, numărul `s[j]` de părinţi ai săi. Cel mai bătrân marţian, adică cel care vorbeşte primul, este acela care nu are nici un părinte în viaţă, adică cel pentru care `s[j]`este `0`. Il scriem în fişier şi apoi, pentru fiecare copil al său, micşorăm cu `1` numărul de părinţi deoarece părintele `j` a fost deja pus în lista vorbitorilor şi de el nu trebuie să mai ţinem cont în continuare. Punem `-1` în `s[j]` pentru o eventuală depanare (ar fi suficient să punem o valoare diferită de `0`).

```
program marte;
var n,i,j,k:integer;
    a:array[1..100,1..100] of integer;
    f,g:text;
    s:array[1..100] of integer;
begin
assign(f,'date.in'); reset(f);
assign(g,'date.out'); rewrite(g);
readln(f,n);

for i:=1 to n do
      repeat
           read(f,j);
           if j>0 then a[i,j]:=1;
      until j=0;
close(f);
for j:=1 to n do
     begin
         s[j]:=0;
         for i:=1 to n do
              s[j]:=s[j]+a[i,j];
     end;
for i:=1 to n do
     begin
          j:=1;
```

```
          while s[j]<>0 do  j:=j+1;
          write(g,j,' ');
          s[j]:=-1;
          for k:=1 to n do
                   if a[j,k]=1 then s[k]:=s[k]-1;
     end;
close(g);
end.
```

Tehnoredactare

Roşca Valentin