Buzău, 2-9 aprilie 2004

Olimpiada NAȚIONALĂ de informatică 2004

Vă prezentăm în continuare enunțurile celor 18 de probleme propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică pentru liceeni, urmând ca în numărul următor să publicăm și problemele propuse spre rezolvare la gimnaziu și pentru lărgirea lotului național de informatică.

Clasa a IX-a

P040411: Coduri

Un detectiv particular are de rezolvat un caz special. Este vorba de o deturnare de fonduri. Pentru a putea rezolva cazul trebuie să găsească un șir cu n coduri distincte. Fiecare cod este un număr natural scris în baza 10. Din păcate lucrurile nu sunt simple, pentru că din cercetările efectuate a obținut două informații. Prima informație este legată de faptul că suma pătratelor codurilor este un cub perfect, iar a doua spune că suma cuburilor codurilor este un pătrat perfect.

Ajutați detectivul să găsească un șir de coduri x_1, x_2, …, x_n, care verifică condițiile din enunț.

Date de intrare

Fișierul de intrare **coduri.in** conține pe prima linie numărul natural n.

Date de ieșire

Fișierul de ieșire **coduri.out** va conține n linii, câte una pentru fiecare cod din șir, în ordine crescătoare.

Restricții

- $1 \le n \le 20$;
- $x_i \le n^{14}$, $\forall$ 1 ≤ i ≤ n.

Exemplu

coduri.in	coduri.out
2	625
	1250

Timp de execuție: 1 secundă/test

P040412: Logic

Demonstrarea automată a teoremelor și verificarea satisfiabilității unei formule constituie două capitole importante în cadrul logicii matematice. Formulele propoziționale sunt alcătuite din variabile propoziționale (variabile care pot lua doar două valori: *adevărat* sau *fals*) și din operatorii logici *și*, *sau*, *negație*, *echivalent*, *implică*.

Iată câteva exemple de formule propoziționale:

```
~p&(q<=>p)=>q
p|q<=>~p&~q
p
p=>q=>a=>t=>~p
```

În acest exemplu, `p`, `q`, `t` și `a` sunt variabilele propoziționale, `~` este operatorul unar *negație*, `&` este operatorul binar *și*, `|` este operatorul binar *sau*, `=>` este *implicația* logică (apare numai în acest sens, nu apare și `<=`), iar `<=>` este *echivalența* logică. În plus, într-o formulă propozițională pot să apară și paranteze care stabilesc ordinea operațiilor. În lipsa parantezelor, operatorii, în ordinea priorității lor, sunt: `~`, `&`, `|`, `=>` și `<=>`.

În formulele de forma "$A_1 op A_2 op ... op A_K$" asocierile se fac de la dreapta la stânga (adică "$A_1 op (A_2 op (... op A_K) ...)$"), unde *op* este unul dintre `&`, `|`, `=>` sau `<=>` și A_i sunt formule propoziționale, cu i de la 1 la K.

În general, o formulă propozițională se definește astfel:

- orice variabilă propozițională este formulă propozițională;
- dacă `A` și `B` sunt formule propoziționale, atunci `(A)`, `~A`, `A&B`, `A|B`, `A=>B` și `A<=>B` sunt formule propoziționale.

Dacă înlocuim într-o formulă propozițională toate variabilele cu valori de adevăr (*adevărat* sau *fals*), obținem o afirmație. Valoarea de adevăr a unei afirmații este dată de următoarea definiție:

- dacă afirmația constă dintr-o singură valoare de adevăr, afirmația ia valoarea respectivă;
- dacă `A` și `B` sunt afirmații, atunci:
  - `A` este adevărată dacă și numai dacă valoarea sa de adevăr este *adevărat*;

- `(A)` este adevărată dacă și numai dacă `A` este adevărată;
- `~A` este falsă dacă și numai dacă `A` este adevărată;
- `A&B` este adevărată dacă și numai dacă atât `A`, cât și `B` sunt adevărate;
- `A|B` este falsă dacă și numai dacă `A` este fals și `B` este fals;
- `A=>B` este adevărată dacă și numai dacă `~A|B` este adevărată;
- `A<=>B` este adevărată dacă și numai dacă `(A=>B)&(B=>A)` este adevărată.

Se numește soluție a formulei propoziționale P (formulă în care apar numai variabilele propoziționale distincte A_1, A_2, ..., A_N) orice N-uplu (v_1, v_2, ..., v_N) (cu v_i valori de adevăr) pentru care înlocuind fiecare variabilă A_i cu valoarea v_i, afirmația rezultată este adevărată.

Logica fiind un obiect nesuferit de studenții de la informatică, ei apelează la informaticienii din clasa a IX-a pentru a-i ajuta să numere câte soluții distincte are o formulă propozițională dată.

Date de intrare
În fișierul de intrare **`logic.in`** se găsește pe prima linie o formulă propozițională în care variabilele propoziționale sunt reprezentate de litere mici ale alfabetului englez.

Date de ieșire
În fișierul de ieșire **`logic.out`** se va scrie pe prima linie numărul de soluții pentru formula propozițională din fișierul de intrare, urmat de caracterul sfârșit de linie.

Restricții și precizări
- la intrare se va da întotdeauna o formulă propozițională corectă sintactic;
- formula are mai puțin de 232 de caractere;
- în formulă nu apar mai mult de 10 litere mici ale alfabetului latin;
- se recomandă ca afișarea soluției să se facă în modul următor:
  - *Pascal*: `writeln(f, nsol);`
  - *C*: `fprintf(f, "%d\n", nsol);`
  - *C++*: `f << nsol << "\n";`

Exemple

`logic.in`	**`logic.out`**
`p|q<=>~p&~q`	`4`

`logic.in`	**`logic.out`**
`a`	`1`

Timp de execuție: 1 secundă/test

P040413: Poligon
Se consideră un caroiaj cu dimensiunea $M \times N$ în care sunt plasate K puncte. Fiecare punct poate fi legat de vecinul său direct pe cel mult opt direcții (*N*, *NE*, *E*, *SE*, *S*, *SV*, *V*, *NV*).

Determinați patrulaterele având vârfurile în punctele date, iar laturile formate din legături între două sau mai multe puncte coliniare.

Date de intrare
Pe prima linie a fișierului de intrare **`poligon.in`** se află trei numere întregi M, N și K, separate prin spațiu, care reprezintă dimensiunile caroiajului și numărul de puncte.

Fiecare dintre următoarele K linii conține câte trei numere naturale nenule I, J și V, separate printr-un singur spațiu, care reprezintă coordonatele unui punct (perechea (I, J)), respectiv direcțiile pe care acest punct poate fi legat de vecinii direcți.

Codificarea direcțiilor se face printr-un număr cuprins între `0` și 255. Reprezentarea binară a acestuia, pe 8 cifre semnifică, începând de la stânga spre dreapta, legătură pe direcțiile: *N NE E SE S SV V NV*. Valoarea `0` indică faptul că punctul nu poate fi conectat de vecinul direct de pe direcția respectivă.

De exemplu:

$(1\ 0\ 0\ 0\ 0\ 1\ 1\ 0)_2 = 134_{10}$, deci, legături spre *N*, *SV*, *V*.

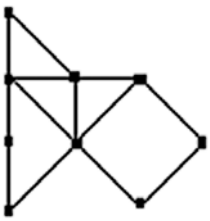
Date de ieșire
Fișierul de ieșire **`poligon.out`** va conține pe o singură linie un singur număr natural care reprezintă numărul de patrulatere care respectă condiția din enunț.

Restricții
- $1 < M, N \leq 200$;
- $4 \leq K \leq 50$.

Exemplu

`poligon.in`
```
4 4 9
1 1 24
2 1 184
2 2 43
2 3 22
3 1 136
3 2 213
3 4 5
4 1 192
4 3 65
```

`poligon.out`
```
6
```



Timp de execuție: 2 secunde/test

P040414: Șablon
Gigel și *Vasilică* își imaginează un mod de a transmite mesaje pe care nimeni să nu le poată descifra.

Mesajul este ascuns într-un text care are N linii și pe fiecare linie sunt exact N caractere - litere mari ale alfabetului englez, cifre, semne de punctuație și caracterul spațiu.

Decodificarea se face cu ajutorul unui șablon, de aceleași dimensiuni ca și textul, care are câteva găuri. Suprapunând șablonul peste text rămân vizibile câteva caractere. Acestea se citesc în ordinea liniilor, de sus în jos, iar pe aceeași linie de la stânga la dreapta. Apoi hârtia cu textul se rotește spre stânga, în sens trigonometric, cu 90°, șablonul rămânând fix. Alte caractere devin vizibile și acestea se citesc în același mod. Operația se repetă de încă două ori (rotire cu 180°, respectiv cu 270°), până când textul ajunge, printr-o nouă rotație cu 90°, din nou în poziția inițială.

Din păcate, șablonul pentru codificare/decodificare s-a pierdut. În schimb a rămas la *Gigel* mesajul inițial, iar la *Vasilică* a ajuns textul care conține mesajul.

Se cere să se reconstituie șablonul care a fost folosit la codificare.

Date de intrare

Fișierul de intrare **sablon.in** conține pe prima linie mesajul inițial. Pe linia a doua a fișierului de intrare se găsește valoarea *N*. Următoarele *N* linii conțin textul care ascunde mesajul.

Date de ieșire

Fișierul de ieșire **sablon.out** trebuie să conțină *N* linii a câte *N* caractere. Caracterele pot fi "O" (pentru reprezentarea unei găuri) și "X".

Restricții și precizări

- prin rotirea textului nici una din găuri nu se va suprapune peste nici una din pozițiile ocupate de o gaură în pozițiile precedente ale textului;
- $1 \le N \le 50$.
- mesajul are cel mult 1000 caractere și se încheie cu un caracter diferit de spațiu;
- în cazul în care există mai multe soluții, afișați una dintre ele.

Exemplu

sablon.in

```
CODIFICARE CU SABLON
10
ABCDCEFAGH
IJOKLEMNOP
DQRSTUVWCX
YZAIBCRDEF
GFHIJKLMNI
AJKLMNOPSQ
RSTOUV WXY
ZBABCDEFGU
HIJKNLMCNO
PQLRS TUVW
```

sablon.out

```
XXXXOXXXXX
XXOXXXXXXX
OXXXXXXXXX
XXXOXXXXXX
XOXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```

Timp de execuție: 1 secundă/test

P040415: Șir

Gigel se distrează construind șiruri crescătoare de numere din mulțimea {1, 2, …, *n*}. La un moment dat observă că unele șiruri, de cel puțin *k* termeni ($k \ge 3$), au o proprietate mai aparte: diferența dintre doi termeni consecutivi este constantă. Iată câteva exemple de astfel de șiruri pentru $n \ge 22$:

2, 3 ,4
1, 5, 9, 13
7, 10, 13, 16, 19, 22

Dându-se numărul natural *n* ajutați-l pe *Gigel* să numere câte astfel de șiruri poate să construiască.

Date de intrare

În fișierul de intrare **sir.in** se găsește, pe prima linie, numărul *n*.

Date de ieșire

În fișierul de ieșire **sir.out** se va afișa, pe prima linie, numărul cerut urmat de caracterul sfârșit de linie.

Restricții și precizări

- dacă există mai multe soluții, se va determina una singură;
- $3 \le n \le 20.000$;
- $3 \le k \le n$.

Exemple

sir.in	**sir.out**
3	1

sir.in	**sir.out**
4	3

sir.in	**sir.out**
5	7

Timp de execuție: 1 secundă/test

P040416: Snipers

Se spune că în timpul războiului cu *gnomii*, *trolii* au trimis *n* trăgători de elită să lichideze cele *n* căpetenii inamice.

Din fericire căpeteniile inamice erau plasate în câmp deschis, iar trăgătorii au reușit să se plaseze în zonă fără să fie observați. Când să fie dată comanda de tragere s-a constatat că nu se transmisese fiecărui trăgător ce căpetenie să împuște, iar dacă doi trăgători ar fi tras în aceeași căpetenie

sau traiectoriile razelor ucigașe s-ar fi intersectat, atunci ar fi scăpat cel puțin o căpetenie care ar fi putut duce războiul până la capăt, iar *trolii* ar fi fost învinși. Deoarece căpeteniile aveau capacitatea de a deveni invizibile oricând doreau (pe o perioadă nelimitată), trebuiau lichidate simultan, altfel… Istoria ne spune că *trolii* au învins deoarece comandantul lor a reușit ca în mai puțin de o secundă să transmită fiecărui trăgător în ce căpetenie să tragă. Voi puteți face asta?

Scrieți un program care, citind pozițiile trăgătorilor și a căpeteniilor, determină căpetenia în care trebuie să tragă fiecare trăgător.

Date de intrare

Fișierul de intrare **snipers.in** conține pe prima sa linie numărul n. Pe următoarele n linii se află perechi de numere întregi, separate prin spațiu, care reprezintă coordonatele trăgătorilor urmate de alte n perechi de numere întregi care reprezintă coordonatele căpeteniilor (abscisă și ordonată).

Date de ieșire

Fișierul de ieșire **snipers.out** va conține n linii. Linia i a fișierului va conține numărul căpeteniei țintite de trăgătorul i ($1 \le i \le n$).

Restricții și precizări

- $0 < n < 200$;
- coordonatele sunt numere întregi cuprinse între 0 și 50000;
- raza ucigașă a oricărei arme se oprește în ținta sa;
- în datele de intrare nu vor exista trei persoane aflate în puncte coliniare.

Exemple

snipers.in
```
2
1 3
1 1
3 4
3 1
```

snipers.out
```
1
2
```

snipers.in
```
5
6 6
4 13
2 8
9 4
5 2
6 11
9 7
3 9
1 4
7 3
```

snipers.out
```
2
1
3
4
5
```

Timp de execuție: 1 secundă/test

Clasa a X-a

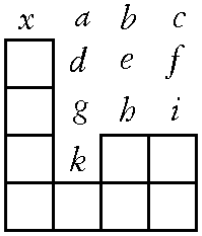
P040417: Găina

Găina *Chucky* 767 trebuie să străbată curtea sărind de pe un coteț pe altul, sau zburând peste cotețe, de la primul până la ultimul coteț. Cotețele sunt reprezentate prin niște dreptunghiuri de lățime 1 m și înălțimi date și sunt numerotate în ordine de la stânga cu numere de 1 la n. Două cotețe cu numere consecutive sunt lipite (adiacente). Inițial, *Chucky* are un număr de unități de energie. Dacă la un moment dat *Chucky* ajunge să aibă energie strict negativă sau se află în imposibilitatea de a mai face un pas (întâlnește un coteț mai înalt decât cota la care se află), atunci nu mai poate continua acel drum.

Chucky poate să se deplaseze făcând următoarele tipuri de mișcări:

- ***pas*** – găina pășește pe orizontală de pe un coteț de înălțime H pe următorul coteț de aceeași înălțime (în desenul de mai jos: de la poziția h la i); în acest caz nu se pierde și nu se câștigă energie;
- ***aterizare*** – găina aterizează pe un coteț de înălțime H, venind în zbor, tot de la înălțimea H (exemplu desenul de mai jos: de la g la h); nici în acest caz nu se pierde și nu se câștigă energie;
- ***decolare*** – găina zboară 1 m pe orizontală de pe un coteț (în desen de la x la a); în acest caz găina pierde o unitate de energie;
- ***zbor orizontal*** – găina zboară pe orizontală (exemplu în desen, de la a la b, de la b la c, …); în acest caz pierde câte o unitate de energie pentru fiecare metru parcurs pe orizontală.
- ***picaj*** – găina coboară pe verticală (exemplu în desen de la a la d, sau de la d la g …); în acest caz câștigă câte o unitate de energie pentru fiecare metru coborât.

Mai jos, avem un drum format din patru cotețe, de înălțimi 4, 1, 2, 2. Exemplificăm diferite tipuri de mișcări din poziția x (ceea ce nu reprezintă un traseu complet).



În figura anterioară, din poziția de start x se poate ajunge în poziția h pe 3 trasee:

- x, a, b, e, h cu energia inițială 2:
  $x \to a(-1)$, $a \to b(-1)$, $b \to e(+1)$, $e \to h(+1)$
- x, a, b, e, h cu energia inițială 1:
  $x \to a(-1)$, $a \to d(+1)$, $d \to e(-1)$, $e \to h(+1)$
- x, a, d, g, h cu energia inițială 1:
  $x \to a(-1)$, $a \to d(+1)$, $d \to g(+1)$, $g \to h(0)$

Să se determine energia minimă (un număr natural) pe care trebuie să o aibă *Chucky* la începutul călătoriei (când se află pe primul coteț), astfel încât ea să poată ajunge pe cotețul *n*, având în fiecare moment energia mai mare sau egală cu 0.

Date de intrare
Fișierul de intrare **gaina.in** conține două linii. Pe prima linie se află numărul natural *n*. Pe linia a doua se află *n* numere naturale h_1, h_2, ..., h_n (unde h_i reprezintă înălțimea cotețului *i*), separate printr-un spațiu.

Date de ieșire
Fișierul de ieșire **gaina.out** va conține o singură linie pe care se află un număr natural *K* reprezentând energia minimă inițială cu care găina *Chucky* poate să ajungă la cotețul *n*, având în fiecare moment energia mai mare sau egală cu 0.

Restricții și precizări
- $0 < n < 10001$;
- $0 \le h_i \le 30000$;
- se garantează existența unei soluții (primul coteț are înălțimea mai mare sau egală cu înălțimea oricărui alt coteț).

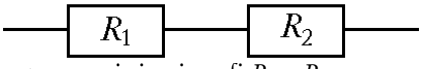
Exemple

gaina.in	gaina.out
3 2 2 0	16

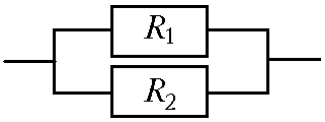
gaina.in	gaina.out
6 3 0 0 2 1 1	2

Timpi de execuție:
- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

P040418: Rez
Gigel este electronist amator. El afirmă că a inventat o nouă componentă electronică denumită *reziston*. În mod ciudat totuși *rezistonii* au niște proprietăți care nouă ne sună foarte cunoscut:
- orice *reziston* este caracterizat printr-o mărime fizică numită *rezistență*; aceasta poate avea ca valori numai numere naturale;
- *rezistonii* pot fi legați între ei în serie sau în paralel, formând astfel circuite;
- fie doi *rezistoni* având *rezistențele* R_1, respectiv R_2; legarea în serie a *rezistonilor* se realizează astfel:



Rezistența acestui circuit va fi $R_1 + R_2$.
- legarea celor doi *rezistoni* în paralel se realizează astfel:



Rezistența acestui circuit va fi $(R_1 \cdot R_2) / (R_1 + R_2)$. Fiindcă *rezistențele* pot fi numai numere naturale, împărțirea este întreagă (adică rezultatul este câtul împărțirii întregi a lui $(R_1 \cdot R_2)$ la $(R_1 + R_2)$).
- prin legarea oricâtor *rezistoni* în serie și în paralel se obțin circuite; circuitele pot fi legate în serie și/sau în paralel după aceleași reguli; *rezistența* unui circuit se calculează aplicând regulile de mai sus.

Un circuit va fi codificat printr-un șir de caractere construit după următoarele reguli:
- dacă circuitul *C* este format dintr-un singur *reziston* și acesta are *rezistența* de valoare *x*, atunci codificarea circuitului *C* este *Rx*; *rezistența* circuitului *C* va fi *x*.
- dacă circuitul *C* este obținut prin legarea în serie a două sau mai multe circuite, codificate C_1, C_2, ..., C_k, atunci codificarea circuitului *C* se obține concatenând în ordine codificările circuitelor $C_1C_2...C_k$; *rezistența* circuitului *C* se obține prin însumarea *rezistențelor* circuitelor C_1, C_2, ..., C_k;
- dacă circuitul *C* este obținut prin legarea în paralel a două sau mai multe circuite, atunci codificarea circuitului *C* se obține încadrând între paranteze rotunde codificările circuitelor din care este format și separând aceste codificări prin virgulă: $(C_1, C_2, ..., C_k)$, $k > 1$; *rezistența* circuitului *C* este egală cu câtul împărțirii produsului dintre *rezistențele* C_1, C_2, ..., C_k și suma *rezistențelor* circuitelor C_1, C_2, ..., C_k.

Scrieți un program care să determine *rezistența* unui circuit.

Date de intrare
Fișierul de intrare **rez.in** conține pe prima linie un șir de caractere care reprezintă codificarea unui circuit conform regulilor de mai sus.

Date de ieșire
Fișierul de ieșire **rez.out** conține o singură linie pe care este scrisă *rezistența* circuitului specificat în fișierul de intrare.

Restricții și precizări
- lungimea codificării unui circuit este un număr întreg cuprins între 0 și 1000, exclusiv;
- *rezistența* oricărui *reziston* este un număr întreg cuprins între 0 și 100, exclusiv;
- *rezistența* oricărui circuit este un număr întreg cuprins între 0 și 2.000.000.000, exclusiv;
- șirul prin care se codifică un circuit nu conține spații;
- pentru datele de test nu se vor obține împărțiri la 0.

Exemple

rez.in	rez.out
R12	12

rez.in	rez.out
R42R33R3	78

rez.in	rez.out
R2(R5,R69,R12)R80	130

rez.in	rez.out
(R5,R3(R12,R4),R3)	6

Timpi de execuție:
• *Microsoft Windows*: 0,1 secunde/test
• *Linux*: 0,1 secunde/test

P040419: Sortări

Balaurului Arhirel nu îi plac prea mult șirurile care nu sunt ordonate. Din acest motiv, nu poate să suporte permutările de N elemente, așa că se decide să le sorteze și pentru asta inventează o metodă proprie de sortare.

El ia inițial un șir S care reprezintă o permutare de ordin N. Caută în șirul S cel mai mic (*min*) și cel mai mare element (*max*). Să considerăm că *min* se află în șirul S pe poziția *pmin*, iar *max* pe poziția *pmax*. Să notăm cu x minimul dintre *pmin* și *pmax*, iar cu y maximul dintre *pmin* și *pmax*. Șirul S a fost astfel partiționat în alte trei șiruri S_1, S_2, S_3 care pot avea fiecare zero elemente, un element sau mai multe elemente. Șirul S_1 începe la prima poziție din șir și se termină la poziția x - 1. Șirul S_2 începe la poziția x + 1 și se termină la poziția y - 1. Șirul S_3 începe la poziția y + 1 și se termină la ultima poziție din șir.

Balaurul Arhirel mută valoarea *min* la capătul din stânga al lui S, iar valoarea *max* la capătul din dreapta al șirului S și reia sortarea pentru fiecare din șirurile S_1, S_2 și S_3.

De exemplu, să considerăm N = 6 și șirul S = (3 4 2 1 6 5). La primul pas *min* = 1 și *max* = 6. Deci S_1 = (3 4 2); S_2 = (); S_3 = (5). Se mută *min* și *max* la capetele șirului și se obține S = (1 3 4 2 5 6) și se sortează în același mod S_1, S_2 și S_3. S_2 și S_3 au 0, respectiv 1 element, deci sunt deja sortate. Pentru S_1, se găsește *min* = 2 și *max* = 4 și vom avea șirurile (3), () și (). Se mută *min* și *max* la capete și se obține S_1 = (2 3 4). În final, vom avea șirul S = (1 2 3 4 5 6).

Evident, această metodă nu va funcționa întotdeauna pentru sortarea unei permutări. Spre exemplu, pentru șirul S = (3 4 1 6 5 2), se găsește *min* = 1 și *max* = 6, iar S_1 = (3 4), S_2 = (), S_3 = (5 2). Se mută *min* și *max* la capetele lui S: S = (1 3 4 5 2 6) și se procedează la sortarea pe rând a șirurilor S_1, S_2, S_3. S_1 este sortat, S_2 nu are elemente, iar S_3 va deveni S_3 = (2 5). În final, S = (1 3 4 2 5 6).

Ajutați-l pe *Balaurul Arhirel* să afle câte dintre permutările de N elemente pot fi sortate prin metoda sa.

Date de intrare
Fișierul **sortari.in** conține o singură linie pe care se află numărul N.

Date de ieșire
Fișierul **sortari.out** va conține o singură linie pe care se află numărul de permutări de ordin N ce pot fi sortate prin metoda balaurului modulo 19573 (restul împărțirii numărului de permutări la 19573).

Restricție
• 0 < N < 201.

Exemplu

sortari.in	sortari.out
4	18

Explicație
În cazul permutărilor de câte patru elemente, șirurile (1 3 4 2); (3 1 2 4); (3 1 4 2); (3 4 1 2); (3 4 2 1); (4 3 1 2) nu vor putea fi sortate crescător. Celelalte 24 - 6 = 18 permutări pot fi sortate crescător.

De exemplu, pentru șirul (1 3 4 2), după găsirea *min* = 1, *max* = 4, se obțin șirurile: S_1 = (); S_2 = (3); S_3 = (2), care, având câte un element sunt sortate. Astfel, după mutarea la capete a lui *min* și *max* vom avea: (1 3 2 4)

Timpi de execuție:
• *Microsoft Windows*: 1,5 secunde/test
• *Linux*: 0,2 secunde/test

P040420: Cuvinte

Balaurul Arhirel se decide să învețe biologie, așa că dorește să cumpere manualul de clasa a X-a. Din păcate, acesta nu se mai găsește pe piață, dar *Balaurul* reușește să găsească o copie la un prieten.

După ce începe să citească, *Balaurul Arhirel* observă că există greșeli în copia prietenului, iar într-un impuls de energie, se hotărăște să corecteze manualul.

El are la dispoziție un dicționar de M cuvinte dintre care trebuie să extragă variante pentru cuvântul greșit. Asupra cuvântului greșit balaurul poate să facă următoarele schimbări în așa fel încât acesta să ajungă la o variantă din dicționar:
• poate șterge o literă;
• poate insera o literă;
• poate schimba o literă în altă literă.

Totuși, *Balaurul Arhirel* este leneș, așa că nu dorește să opereze mai mult de K schimbări în cuvântul greșit pentru a-l aduce la o formă corectă (existentă în dicționar).

Ajutați-l pe *Balaurul Arhirel* să afle care dintre cuvintele din dicționar ar putea fi variante ale cuvântului greșit.

Date de intrare
Fișierul de intrare **cuvinte.in** conține pe prima linie cele două numere M și K, separate printr-un spațiu, reprezentând numărul de cuvinte din dicționar și numărul maxim de modificări care pot fi efectuate asupra cuvântului care trebuie corectat.

Pe cea de-a doua linie se găsesc separate printr-un spațiu lungimea cuvântului greșit, $L_{cuvânt}$, și cuvântul greșit.

Pe următoarele M linii se găsesc cuvintele din dicționar, câte un cuvânt pe o linie în forma următoare: pe linia i lungimea L_{i-2} a cuvântului i - 2, separată printr-un singur spațiu de cuvântul i - 2.

Date de ieșire

Fișierul de ieșire **cuvinte.out** va conține M linii. Pe linia i se află valoarea 1 pentru cazul în care cuvântul i din dicționar este o variantă pentru cuvântul greșit dat, respectiv valoarea 0 în caz contrar.

Restricții și precizări

- $0 < M < 21$;
- $0 < K < 31$;
- lungimea oricărui cuvânt (inclusiv greșit) este un număr întreg cuprins între 0 și 10001, exclusiv;
- cuvintele sunt formate doar din literele alfabetului latin, iar literele mici diferă de cele mari (de exemplu, z nu este același lucru cu z).

Exemplu

cuvinte.in	cuvinte.out
6 2	0
6 radiux	1
5 ladin	0
6 Radius	1
6 ridica	0
5 radio	1
6 adipos	
5 cadiu	

Timpi de execuție:

- *Microsoft Windows*: 2 secunde/test
- *Linux*: 0,3 secunde/test

P040421: Materom

Liceul Național Anonim (*LNA*) este invitat să participe la olimpiada de matematică-română cu o echipă formată din m elevi.

La această olimpiadă elevii lucrează în echipă și trebuie să rezolve două subiecte: unul de română și altul de matematică.

Au fost testați și punctați la cele două materii n elevi, numerotați de la 1 la n.

Așa cum era de așteptat, în general, elevii buni la matematică s-au dovedit a fi cam slăbuți la română și viceversa.

Pentru a maximiza șansele de câștig ale echipei *LNA*, directorul a decis să trimită m elevi dintre cei n elevi testați, astfel încât diferența în modul dintre suma punctajelor de la limba română ale elevilor din echipă și suma punctajelor la matematică ale elevilor din echipă să fie minimă. Dacă există mai multe echipe de elevi care îndeplinesc condiția precedentă, va fi selectată dintre acestea o echipă pentru care suma tuturor notelor să fie maximă.

Scrieți un program care să determine în conformitate cu decizia directorului, diferența în modul dintre suma punctajelor de la limba română ale elevilor din echipa *LNA* și suma punctajelor la matematică ale elevilor din echipă, precum și suma tuturor punctajelor elevilor din echipa *LNA*.

Date de intrare

În fișierul de intrare **materom.in** se află pe prima linie numerele naturale n și m separate printr-un spațiu, având semnificația din enunț.

Pe fiecare dintre următoarele n linii se află două numere naturale separate printr-un spațiu. Mai exact linia i din fișier ($2 \le i \le n + 1$) conține m_i și r_i, unde m_i este punctajul obținut la matematică, iar r_i este punctajul obținut la limba română de elevul i - 1.

Date de ieșire

Fișierul de ieșire **materom.out** conține două linii. Pe prima linie se va afișa diferența (în modul) dintre suma punctajelor de la limba română ale elevilor din echipă și suma punctajelor la matematică ale elevilor din echipă. Pe cea de a doua linie se va afișa suma punctajelor elevilor selectați în echipa *LNA*.

Restricții și precizări

- $1 \le m < 20$;
- $1 \le n \le 500$;
- $m \le n$;
- $0 \le m_i, r_i \le 20$.

Exemplu

materom.in	materom.out
4 2	2
2 3	10
1 2	
6 2	
4 1	

Explicație

Dintre cei patru elevi trebuie să selectăm doi. Avem 6 posibilități, dintre care trei au diferența (în modul) dintre suma notelor la matematică și suma notelor la română 2.

Acestea sunt:

- (1, 2) pentru care suma punctajelor este 8;
- (1, 4) pentru care suma punctajelor este 10;
- (2, 4) pentru care suma punctajelor este 8.

Alegem combinația 1 4, deoarece are suma maximă.

Timpi de execuție:

- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,1 secunde/test

P040422: Puncte

Considerăm că toate punctele de coordonate întregi din plan sunt colorate în negru, cu excepția a n puncte care sunt colorate în roșu. Două puncte roșii aflate pe aceeași linie orizontală sau pe aceeași linie verticală (adică puncte care au aceeași ordonată sau aceeași abscisă) pot fi unite printr-un segment. Colorăm în roșu toate punctele de coordonate întregi de pe acest segment. Repetăm operația cât timp se obțin puncte roșii noi.

Cunoscând coordonatele celor *n* puncte care erau inițial roșii, aflați numărul maxim de puncte roșii care vor exista în final.

Date de intrare
Pe prima linie a fișierului de intrare **puncte.in** se află numărul *n*. Pe următoarele *n* linii sunt date coordonatele punctelor, separate printr-un singur spațiu.

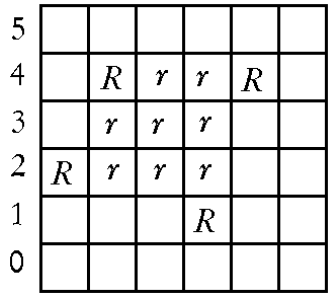
Date de ieșire
Fișierul de ieșire **puncte.out** va conține o singură linie pe care se află numărul maxim de puncte roșii existente în final.

Restricții și precizări
- $0 \le n \le 100.000$;
- coordonatele sunt numere întregi cuprinse între 0 și 1000.

Exemplu

puncte.in
```
4
0 2
3 1
1 4
4 4
```

puncte.out
```
12
```



Timpi de execuție:
- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

Clasele a XI-a și a XII-a

P040423: Color

Ion și *Vasile* joacă un joc. Ei au la dispoziție un arbore binar strict (adică fiecare nod are 0 sau doi fii) cu *N* noduri, numerotate de la 1 la *N* (nodul numerotat cu 1 este rădăcina arborelui). Inițial, toate nodurile sunt colorate în alb. Jucătorii vor efectua mutări alternativ, iar jucătorul aflat la mutare va colora în negru un nod colorat în alb. *Ion* efectuează prima mutare și poate colora în negru orice nod al arborelui. Considerând că ultimul nod colorat de unul dintre jucători este *P*, jucătorul care urmează la mutare poate colora în negru unul din următoarele noduri (dacă nu au fost deja colorate în negru):
- unul din cei doi fii ai lui *P* (dacă *P* nu este frunză în arbore);
- tatăl lui *P* (dacă *P* nu este rădăcina arborelui).

Jocul continuă până când unul dintre jucători nu mai poate efectua nici o mutare. Atunci, jucătorul care a efectuat ultima mutare este considerat câștigător.

Considerând că ambii jucători joacă optim, determinați toate nodurile din arbore pe care le poate colora *Ion* la prima mutare, astfel încât să fie sigur de victorie.

Date de intrare
Prima linie a fișierului **color.in** conține numărul întreg *N*, reprezentând numărul de noduri din arbore.

Următoarele *N* - 1 linii conțin câte două numere întregi separate printr-un spațiu, *a* și *b*, având semnificația că *a* este tatăl lui *b*.

Date de ieșire
Pe prima linie a fișierului **color.out** se va afla numărul întreg *M*, reprezentând numărul de noduri pe care le poate colora *Ion* la prima mutare, astfel încât să fie sigur de victorie. Pe următoarea linie veți afișa numerele acestor noduri, în ordine crescătoare.

Restricție
- $1 \le N \le 16.000$, *N* impar.

Exemplu

color.in
```
9
1 2
1 3
2 4
2 5
4 6
4 7
3 8
3 9
```

color.out
```
6
1 5 6 7 8 9
```

Timpi de execuție:
- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,3 secunde/test

P040424: Magic

Misopan și *Trofonaced* sunt doi eroi care vor să-și unească forțele în lupta împotriva răului. Regatul este reprezentat printr-o matrice dreptunghiulară de *N* linii și *M* coloane. Fiecare element al matricei corespunde unei bucăți de teren uscat sau mlăștinos. Cei doi eroi nu se vor aventura în părțile mlăștinoase ale regatului – se vor deplasa numai pe uscat. Ei se pot muta dintr-o poziție a matricei în una din cele patru poziții vecine pe orizontală sau pe verticală, dacă această poziție corespunde unei zone de uscat. Unele poziții de uscat pot fi transformate prin vrajă în mlaștină.

Ajutați un vrăjitor malefic să aleagă un număr minim de poziții "transformabile", prin schimbarea cărora cei doi eroi să nu se poată întâlni (să nu existe un drum pe uscat între cei doi).

Date de intrare
Prima linie a fișierului **magic.in** conține două numere întregi *N* și *M*, reprezentând numărul de linii, respectiv de coloane ale matricei. Următoarele *N* linii conțin câte *M* caractere cu următoarea semnificație:
- . – poziție uscată;
- x(mic) – poziție mlăștinoasă;

- `*` – poziție uscată "transformabilă" în una mlăștinoasă de către vrăjitor;
- `M` – poziția eroului *Misopan*;
- `T` – pentru poziția eroului *Trofonaced*.

Date de ieșire
Pe prima linie a fișierului **`magic.out`** se scrie numărul întreg *R*, reprezentând numărul minim de poziții care trebuie transformate. Pe următoarele *R* linii vor apărea câte două numere, reprezentând pozițiile alese. Primul număr va fi linia (între 1 și *N*), iar al doilea va fi coloana (între 1 și *M*).

Restricții
- $1 \le N, M \le 50$;
- *R*, rezultatul afișat, poate fi 0;
- se garantează existența unei soluții;
- se garantează că în toată matricea caracterele `M`, respectiv `T` vor apărea fiecare exact o dată;
- pozițiile eroilor sunt implicit zone de uscat care nu pot fi transformate de vrăjitor.

Exemplu

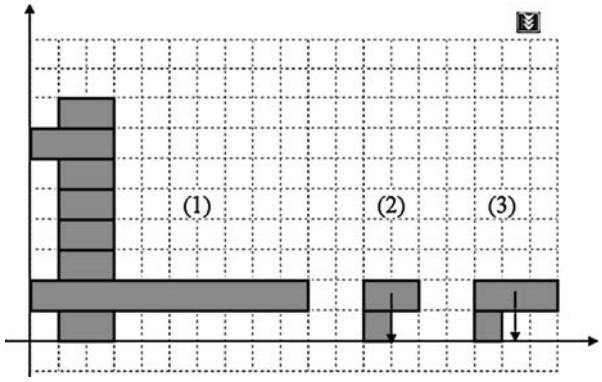
`magic.in`	**`magic.out`**
`4 4`	`1`
`MxxT`	`3 3`
`.x*.`	
`.**.`	
`**x.`	

Timpi de execuție:
- *Microsoft Windows*: 1,8 secunde/test
- *Linux*: 0,6 secunde/test

P040425: Turnuri
Renumitul arhitect *Prăbușilă* dorește să construiască unul dintre cele mai interesante turnuri de pe planetă. Acest turn, în mod cu totul deosebit, va avea etaje de diverse lățimi, între 1 și 100, numere întregi.

Prăbușilă s-a hotărât deja ce dimensiune va avea fiecare dintre etajele turnului, dar nu și cum să le așeze pe orizontală. El ar dori mai întâi să știe câte variante are.

Dacă cele două informații sunt egale, căutarea se termină cu succes.



Etajele pot fi așezate la coordonate întregi și va trebui ca un astfel de turn să nu se dărâme.
- condiția pentru ca un turn să fie stabil este ca la fiecare etaj perpendiculara coborâtă din centrul de greutate al grupului etajelor superioare să cadă strict în interiorul acelui etaj (nu are voie să fie pe margini sau în afară - de exemplu, turnurile 2 și 3 sunt instabile);
- centrul de greutate al unui etaj se află la mijlocul etajului respectiv;
- centrul de greutate al unui grup de etaje are drept coordonată *x* (orizontală) media coordonatelor *x* ale centrelor de greutate ale etajelor componente. (Etajele au mase egale, indiferent de cât de late sunt).

În figura anterioară, etajul din vârf are coordonata *x* a centrului de greutate 2, iar grupul celor două etaje din vârf are centrul de greutate la coordonata *x* = 1,75 (media aritmetică dintre 2 și 1,5)

Se observă în figura anterioară că, deși perpendiculara din centrul de greutate al etajului doi cade în afara etajului unu, totuși turnul este stabil, deoarece perpendiculara din centrul de greutate al grupului format din etajele cuprinse între etajele doi și opt cade strict în interiorul etajului unu.

Să se determine câte turnuri stabile există.

Date de intrare
Fișierul de intrare **`turnuri.in`** conține pe o singură linie lista de numere naturale separate prin câte un spațiu, numere reprezentând lățimile etajelor turnului, începând cu cel mai de sus. Lista se termină cu un 0.

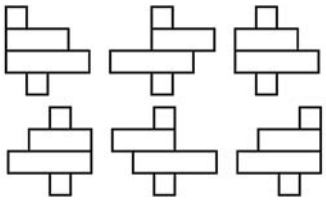
Date de ieșire
Fișierul de ieșire **`turnuri.out`** trebuie să conțină numărul de turnuri.

Restricții și precizări
- numărul maxim de turnuri nu va depăși 2.000.000.000;
- numărul maxim de etaje ale unui turn este 200;
- lățimea maximă a unui etaj este 100.

Exemplu

`turnuri.in`	**`turnuri.out`**
`1 3 4 1 0`	`6`

Explicație
Cele șase variante sunt:



Timpi de execuție:
- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,2 secunde/test

P050326: Coach

Sunteți antrenorul ciclistului *Adirem Onamihs*. În curând va avea loc un eveniment sportiv, iar pentru organizarea acestuia s-au amenajat N intersecții și M drumuri bidirecționale între aceste intersecții. Pentru fiecare drum se cunoaște numărul de minute necesare pentru parcurgerea lui. La fiecare intersecție ciclistul care trece pe acolo este obligat să servească o băutură energizantă și răcoritoare. Băutura diferă de la intersecție la intersecție și se cunoaște deja numărul de calorii ale fiecărei băuturi.

Ca mare antrenor, urmează să întocmiți un plan special pentru a-l antrena pe *Adirem*. Doriți ca durata traseului pe care îl alege *Adirem* să aibă exact T minute, însă nu vreți să-i planuiți întregul traseu (*Adirem* trebuie să își antreneze și mintea, nu numai corpul). Îi veți preciza lui *Adirem* intersecția de unde își începe traseul și intersecția unde îl termină. *Adirem* învață repede – el știe întotdeauna să aleagă traseul optim (drumul cel mai scurt dintre cele două intersecții). Pentru a-l face să meargă exact T minute îi veți interzice trecerea prin anumite intersecții, sub pretextul că valoarea calorică a băuturii servite în intersecția respectivă nu este benefică pentru antrenamentul lui. Astfel, îi veți preciza o limită inferioară și una superioară pentru numărul de calorii ale băuturilor pe care el are voie să le bea. *Adirem* nu va trece decât prin intersecțiile unde se servește o băutură cu valoare calorică între limitele date.

Scrieți un program care să calculeze cele patru variabile din antrenamentul lui *Adirem*: intersecția de start, intersecția de sfârșit, valoarea calorică minimă pe care poate să o consume și valoarea calorică maximă, astfel încât drumul cel mai scurt dintre cele două intersecții (care să respecte restricțiile) să dureze exact T minute.

Date de intrare

Prima linie a fișierului **coach.in** conține trei numere întregi N, M și T – numărul de intersecții, numărul de drumuri, respectiv timpul dorit. Următoarele N linii conțin câte un număr – valorile calorice (întregi între 1 și 10000 inclusiv) ale băuturilor din intersecții, în ordine (de la 1 la N). Următoarele M linii conțin câte un triplet de numere: două intersecții (numere distincte între 1 și N) și durata drumului dintre ele (întreg între 1 și 10000 inclusiv).

Date de ieșire

Fișierul **coach.out** va conține o linie pe care se vor afla cele patru valori găsite: nodul de start, nodul de sfârșit, valoarea calorică minimă și valoarea calorică maximă. Nodurile vor fi întregi între 1 și N, iar valorile calorice vor fi întregi între 1 și 10000 (inclusiv).

Restricții și precizări

- $1 \le N \le 200$;
- $1 \le M \le 4950$;
- $1 \le T \le 1.000.000$;
- intersecțiile găsite (de start și de oprire) trebuie să respecte și ele restricțiile calorice;
- o băutură cu valoare calorică x poate fi băută dacă și numai dacă $c_{min} \le x \le c_{max}$, unde c_{min} și c_{max} sunt valorile calorice minime și maxime stabilite de antrenor;
- între două intersecții există maximum un drum;
- valorile calorice sunt distincte;
- există întotdeauna soluție; dacă există mai multe soluții se cere oricare dintre ele.

Exemplu

coach.in

```
6 9 11
40
10
20
30
60
50
1 2 2
1 3 2
1 4 4
1 6 10
2 3 3
2 4 1
4 5 1
4 6 5
5 6 2
```

coach.out

```
3 6 20 55
```

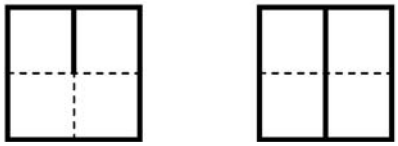
Timpi de execuție:

- *Microsoft Windows*: 0,9 secunde/test
- *Linux*: 0,5 secunde/test

P040427: Pătrate

Ovi este un băiețel foarte isteț căruia îi place să scrie pe asfalt cu creta și să țopăie. El desenează cu cretă roșie un dreptunghi de lățime exact 2 metri și lungime N metri, pe care îl împarte în pătrate egale de latură 1 metru, unele laturi interioare fiind desenate cu cretă roșie, iar restul laturilor interioare cu cretă albă. *Ovi* pornește din pătratul aflat în colțul stânga-sus al dreptunghiului, sărind dintr-un pătrat în altul vecin pe linie sau coloană, cu condiția ca latura care desparte cele două pătrate să nu fie colorată în roșu. El își dorește ca prin sărituri succesive să ajungă în toate pătratele dreptunghiului, dar a observat că numai pentru anumite variante de colorare a laturilor pătratelor reușește acest lucru.

În exemplele de mai jos (cu $N = 2$) liniile interioare îngroșate sunt colorate cu roșu, iar cele punctate sunt colorate cu alb. În exemplul din stânga din figura următoare, pornind din colțul stânga-sus se poate ajunge în oricare alt pătrat, dar în exemplul din dreapta din figura următoare nu se poate ajunge la pătratele din partea dreaptă.



Ajutați-l pe *Ovi* să numere câte posibilități de colorare în roșu a unor laturi interioare ale pătratelor sunt, astfel încât plecând din colțul stânga-sus să poată ajunge prin sărituri în oricare alt pătrat.

Date de intrare
În fișierul **patrate.in** se află un singur număr natural *N* care reprezintă lungimea în metri a dreptunghiului.

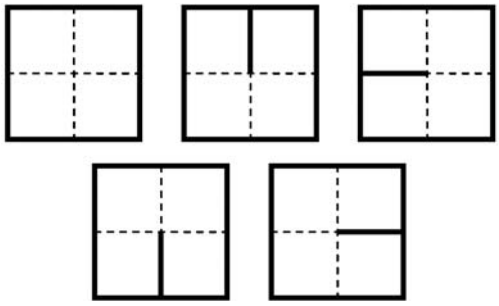
Date de ieșire
În fișierul **patrate.out** se va scrie un singur număr natural (urmat de caracterul de sfârșit de linie) care reprezintă numărul de posibilități cerut.

Restricție
- $1 \le N \le 1000$.

Exemplu
patrate.in
```
2
```

patrate.out
```
5
```

Explicație
Cele cinci posibilități sunt:



Timpi de execuție:
- *Microsoft Windows*: 0,4 secunde/test
- *Linux*: 0,2 secunde/test

P040428: Base3
Se consideră trei numere scrise în baza 3 (folosind cifrele 0, 1 și 2). Se dorește găsirea unui număr *N* în baza 3, care să aibă un număr impar de cifre, iar cifra de pe poziția din mijloc să aibă valoarea 1. Acest număr *N* trebuie obținut prin concatenarea celor trei numere date; în această concatenare, fiecare dintre cele 3 numere poate fi folosit de zero sau mai multe ori.

Determinați numărul minim de cifre pe care îl poate avea un număr având proprietățile precizate mai sus.

Date de intrare
Fișierul de intrare **base3.in** conține trei linii. Pe fiecare linie se află scris un număr în baza 3.

Date de ieșire
Fișierul de ieșire **base3.out** va conține numărul minim de cifre pe care îl poate avea un număr *N* cu proprietățile specificate. Dacă nu se poate obține nici un astfel de număr, se va afișa în fișier valoarea 0.

Restricții
- numărul de cifre al fiecăruia dintre cele trei numere este un număr întreg cuprins între 1 și 16000;
- numerele date pot conține zerouri la început; acestea trebuie luate în considerare, dacă numărul respectiv este folosit în concatenare.

Exemplu

base3.in	base3.out
001 020 2020	13

Explicație
Se poate obține numărul 2020**001**001**001**.

Timpi de execuție:
- *Microsoft Windows*: 0,8 secunde/test
- *Linux*: 0,4 secunde/test

Câștigătorii

Clasa a IX-a
Andrei Ioniță, Suceava - premiul I
Bogdan Ionescu, București - premiul II
Lucian Stănescu, Constanța - premiul III
Teodor Roșu, Vrancea - premiul III
Andrei Gönczi, Arad - premiul III

Clasa a X-a
Dan Spătărel, București - premiul I
Adrian Diaconu, București - premiul II
Adrian Vladu, București - premiul II

Clasa a XI-a
Dan-Leonard Crestez, Brăila - premiul I
Dan-Ionuț Fechete, Suceava - premiul II
Sorin Stancu-Mara, București - premiul III

Clasa a XII-a
Andrei Matei, București - premiul I
Andrei Homescu, Gorj - premiul II
Dan Ghinea, București - premiul III