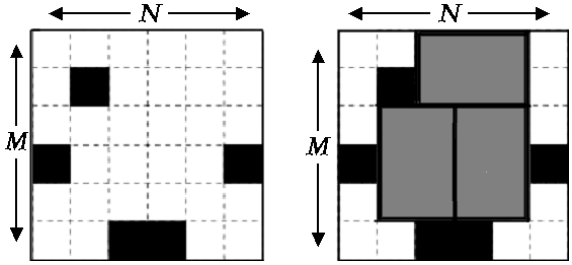
Košice, Slovacia

CEOI 2002

Cea de-a noua ediție a Olimpiadei de Informatică a Europei Centrale s-a desfășurat în perioada 30 iunie - 6 iulie 2002 la Košice, în Slovacia. În continuare veți găsi enunțurile celor șase probleme propuse spre rezolvare la această competiție.

P060207: Bugs Integrated, Inc.

Bugs Integrated, Inc. este o întreprindere specializată în producerea *chip*-urilor de memorie. Firma se pregătește să înceapă producția unor noi *chip*-uri (*Q-RAM*) cu capacitatea de șase teraocteți. Fiecare *chip* este format din șase pătrățele, având latura egală cu unitatea, aranjate în dreptunghiuri de dimensiuni 2×3. *Chip*-urile *Q-RAM* sunt fabricate folosind o placă de silicon de formă dreptunghiulară împărțită în $N \times M$ pătrățele, având latura egală cu unitatea. Toate pătrățelele sunt testate cu atenție și cele defecte sunt marcate cu negru.



Urmează ca din placa de silicon să se taie *chip*-urile. Fiecare *chip* este format din 2×3 (sau 3×2) pătrățele cu latura egală cu unitatea. Bineînțeles, *chip*-urile nu pot conține pătrățele defecte. Este posibil ca prin tăierea plăcii să nu se folosească toate pătrățelele nemarcate. Firma ar dori să piardă cât mai puține pătrățele nemarcate. În concluzie, ei doresc să afle modalitatea de tăiere a plăcii, astfel încât să rezulte un număr maxim de *chip*-uri.

Se consideră dimensiunile a mai multor plăci de silicon și pentru fiecare dintre acestea lista tuturor pătrățelelor defecte. Scrieți un program care calculează pentru fiecare placă numărul maxim de *chip*-uri care pot fi tăiate din acestea.

Date de intrare

Prima linie a fișierului de intrare **BUGS.IN** conține un singur număr întreg D ($1 \le D \le 5$), reprezentând numărul plăcilor de silicon.

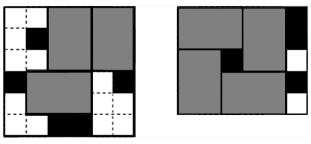
Urmează D blocuri de date, corespunzând celor D plăci de silicon. Prima linie a fiecărui bloc conține trei numere întregi N ($1 \le N \le 350$), M ($1 \le M \le 10$) și K ($0 \le K \le M \times N$) separate printr-un singur spațiu. N reprezintă dimensiunea pe orizontală, corespunzătoare coordonatei x (numărul coloanelor) în care s-a împărțit placa, M reprezintă dimensiunea pe verticală, corespunzătoare coordonatei y (numărul liniilor), iar K este numărul pătrățelelor defecte. Următoarele K linii conțin lista pătrățelelor defecte. Pe fiecare linie se află două numere întregi x și y ($1 \le x \le N$, $1 \le y \le M$) care reprezintă coordonatele a câte unui pătrățel defect (colțul din stânga-sus are coordonatele [1, 1], colțul din dreapta-jos are coordonatele [N, M]).

Date de ieșire

Corespunzător fiecărei plăci descrise în fișierul de intrare, se va scrie în fișierul de ieșire **BUGS.OUT** un singur număr, reprezentând numărul maxim de *chip*-uri care pot fi tăiate din placa respectivă.

Exemplu

BUGS.IN	BUGS.OUT
2 6 6 5 1 4 4 6 2 2 3 6 6 4 6 5 4 3 3 6 1 6 2 6 4	3 4



Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 16 MB

P060208: Batalionul Cuceritorului

În istoria omenirii există câteva bătălii ciudate, cum este și cea care a avut loc în *Franța*, în 1747. În satul *Bassignac-le-Haut*, situat pe malul stâng al râului *Dordogne*, deasupra

digului *Chastang* a existat o cetate. De la dig spre cetate erau amplasate trepte imense. Într-o bună zi s-a apropiat de cetate un batalion condus de *Cuceritor*.

Cuceritorul a fost întâmpinat în dreptul cetății de șeful gărzii de apărare. Acesta, având la dispoziție doar o mică parte din garda sa, i-a spus *Cuceritorului*: "*Văd că ai mulți soldați, așezați în spatele tău pe trepte. Îți propun următorul joc: la fiecare pas îți vei împărți soldații în mod arbitrar în două grupuri. Apoi, eu voi hotărî care dintre cele două grupuri va putea să rămână și care va trebui să plece acasă. Fiecare soldat care rămâne se va urca pe treapta următoare. Dacă cel puțin un soldat de-al tău ajunge pe treapta cea mai înaltă, te voi declara învingător, altfel vei fi învins. Și în acest caz te voi distruge...*"

Cuceritorului i-a plăcut ideea, a acceptat condițiile și a început "cucerirea". Vei avea rolul *Cuceritorului*; există N trepte până la cetate ($2 \le N \le 2000$) și ai cel mult 1000000000 de soldați. Corespunzător fiecărei trepte, se dă numărul de soldați care, în momentul începerii jocului, se află pe treapta respectivă. Treptele sunt numerotate de la 1 (corespunzând treptei cele mai înalte) la N care este treapta cea mai joasă. În momentul începerii jocului, pe treapta având numărul de ordine 1, nu va sta nici un soldat.

Pentru fiecare configurație inițială de joc dată, în cazul în care există strategie de câștig, (adică, poți câștiga indiferent de mutările adversarului), programul tău va trebui să câștige. În caz contrar, programul trebuie să joace corect și să piardă.

Aceasta este o problemă interactivă; vei juca împotriva unei biblioteci. La fiecare pas programul tău va specifica un grup de soldați. Biblioteca va returna `1` sau `2` specificând grupul de soldați care va rămâne (`1` reprezintă grupul specificat de program, iar `2` reprezintă grupul format din restul soldaților). Dacă jocul se sfârșește (fie din cauza că ai câștigat, fie deoarece nu mai există soldați în joc), biblioteca va încheia execuția programului tău în mod corect. Execuția programului nu se poate încheia în nici un alt mod.

Interfața bibliotecii

Biblioteca `libconq` furnizează două subprograme:

- `start`: returnează numărul N al treptelor și încarcă tabloul `stairs` cu numărul soldaților care stau pe ele (valoarea `stairs[i]` este egală cu numărul soldaților care stau pe treapta i);
- `step`: are ca parametru tabloul `subset` (având cel puțin N elemente pentru *FreePascal* și cel puțin $N + 1$ elemente pentru *C/C++*), reprezentând grupul de soldați descris de tine, și returnează `1` sau `2` conform descrierii de mai sus; grupul de soldați este specificat de numărul soldaților aflați pe fiecare treaptă, la fel ca în cazul funcției `start` (valoarea `subset[i]` este egală cu numărul soldaților aparținând primului grup care se află pe treapta i).

Dacă greșești în precizarea unui grup de soldați, jocul se va termina și *Cuceritorul* va fi învins. Treptele sunt numerotate începând cu 1 atât pentru *FreePascal*, cât și pentru *C/C++*.

Declarațiile subprogramelor în FreePascal

```
procedure start(var N:Longint; var stairs:
                                array of Longint);
function step(subset:array of Longint):
                                          Longint;
```

Declarațiile subprogramelor în C/C++

```
void start(int *N,int *stairs);
int step(int *subset);
```

Exemple de utilizare a bibliotecii

Urmează exemple de utilizare a bibliotecii în *FreePascal*, respectiv în *C/C++*; cele două fragmente de program au același efect: pornesc jocul și joacă o rundă, alegând aleator treptele pe care va sta fiecare soldat. Programul tău va trebui să cicleze la infinit.

Exemplu pentru FreePascal:

```
uses libconq;
var stairs:array[1..2000] of Longint;
    subset:array[1..2000] of Longint;
    i,N,result:Longint;
...
start(N,stairs);
...
for i:=1 to N do
  if random(2)=0 then
    subset[i]:=0
  else
    subset[i]:=stairs[i];
  result:=step(subset);
...
```

Exemplu pentru C/C++:

```
#include "libconq.h"
int stairs[2001];
int subset[2001];
int i,N,result;
...
start(&N,stairs);
...
for (i=1;i<=N;i++)
  if (rand()%2==0)
    subset[i]=0;
  else
    subset[i]=stairs[i];
result=step(subset);
...
```

În *Pascal* vei utiliza **`uses`** `libconq`; în *C/C++* vei folosi *`#include "libconq.h"`* și va trebui să comunici compilatorului și argumentul **`libconq.c`**.

Exemplu de comunicare cu biblioteca

Program	*Bibliotecă*
`start(N,stairs)`	$N = 8$, `stairs=(0,1,1,0,3,3,4,0)`
`step((0,1,0,0,1,0,1,0))`	*returnează* 2
`step((0,1,0,0,0,1,0,0))`	*returnează* 2
`step((0,0,0,3,2,0,0,0))`	*returnează* 1
`step((0,0,2,0,0,0,0,0))`	*returnează* 2
`step((0,1,0,0,0,0,0,0))`	*returnează* 2
`step((0,1,0,0,0,0,0,0))`	*nu returnează nimic; ai câștigat*

Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 16 MB

P060209: Un gărduleț decorativ

Richard tocmai a terminat construcția noii sale case. Singurul lucru care mai lipsește este un gărduleț decorativ de lemn. Dar, ca de obicei, habar nu are cum să-l conceapă, deci a decis să comande gardulețe de la o firmă. Întâmplător a găsit un catalog 2002 a firmei *ACME Fence*. După studierea acestuia și-a dat seama ce înseamnă de fapt un gărduleț decorativ.

Un asemenea gărduleț este format din N scânduri, așezate în linie, unele lângă altele, în poziție verticală. Gărdulețul va arăta bine doar dacă îndeplinește următoarele condiții:

- scândurile au lungimi diferite și anume 1, 2, ..., N unități;
- orice scândură care are doi vecini este fie mai lungă decât cei doi vecini ai săi, fie este mai scurtă decât aceștia (observați că astfel vârfurile scândurilor formează o linie frântă care fie urcă, fie coboară de la scândură la scândură);

Rezultă că fiecare gărduleț format din N scânduri poate fi descris în mod unic ca fiind o permutare a_1, ..., a_N a numerelor 1, ..., N, astfel încât:

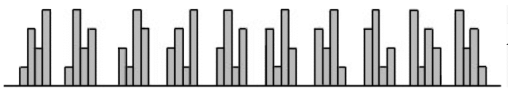
$$\forall i;\ 1 < i < N: (a_i - a_{i-1}) \cdot (a_i - a_{i+1}) > 0.$$

De asemenea, oricare astfel de permutare descrie un gărduleț.

Este evident că astfel, din N scânduri se pot forma multe gărdulețe diferite. Pentru a pune în catalog diferitele modele de gărdulețe într-o oarecare ordine, directorul de vânzări a firmei *ACME* a decis să le numeroteze în felul următor: gărdulețul A (reprezentat de permutarea a_1, ..., a_N) se află în catalog în fața gărdulețului B (reprezentat de permutarea b_1, ..., b_N) dacă și numai dacă există o valoare i, pentru care avem:

$$\forall j < i: a_j = b_j \text{ și } a_i < b_i.$$

De asemenea, pentru a decide care din cele două gărdulețe se va afla mai în față în catalog, se iau permutările corespunzătoare lor, se caută prima poziție în care ele diferă și se compară valorile aflate pe aceste poziții. Toate gărdulețele care se pot forma din N scânduri sunt numerotate (pornind de la 1) în ordinea apariției în catalog. Acest număr îl numim ***număr de catalog***.



Toate gărdulețele formate din N = 4 scânduri, ordonate pe baza numărului de catalog.

După ce *Richard* a studiat cu atenție toate gărdulețele, a decis să comande câteva ditre ele. Pentru fiecare model el și-a notat numărul de scânduri din care acesta este format, precum și numărul de catalog al acestuia. Într-o zi, a fost vizitat de prietenii săi și a vrut să le arate forma gărdulețelor comandate. Dar, zăpăcit cum era de obicei, el a rătăcit catalogul. Ajutați-l să deducă din datele pe care și le-a notat forma gărdulețelor comandate.

Date de intrare

Prima linie a fișierului de intrare **`FENCE.IN`** conține un număr întreg K ($1 \le K \le 100$), reprezentând numărul testelor conținute în fișier. Următoarele K linii conțin fiecare câte un set de date.

Pe fiecare din cele K linii se află două numere întregi N ($1 \le N \le 20$) și C, separate printr-un singur spațiu. N reprezintă numărul scândurilor din gărduleț, iar C este numărul de catalog al gărdulețului respectiv.

Se garantează că numărul gărdulețelor care se pot forma cu 20 de scânduri este un număr întreg care se poate reprezenta pe 64 de biți (**`long long`** în *C/C++*, `int64` în *FreePascal*). De asemenea, puteți fi siguri că fișierul de intrare este corect, mai exact valoarea cea mai mică a lui C este 1 și nu depășește numărul gărdulețelor care se pot forma din N scânduri.

Date de ieșire

Corespunzător fiecărui set de date din fișierul de intrare scrieți în fișierul de ieșire **`FENCE.OUT`** permutarea a_1, ..., a_N corespunzătoare gărdulețului format din N scânduri care are numărul de catalog C. Elementele permutării se vor scrie pe o singură linie și se vor despărți prin câte un spațiu.

Exemplu

`FENCE.IN`	**`FENCE.OUT`**
`2`	`1 2`
`2 1`	`2 3 1`
`3 3`	

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

P060210: Autostrada și cei șapte pitici

A fost odată ca niciodată o țară în care trăiau mai multe familii de pitici. În *Țara Piticilor* fiecare familie locuia în câte o casă. Piticii își vizitau des prietenii din alte familii. Deoarece în *Țara Piticilor* nu exista nici un fel de conflict, se întâmpla frecvent ca toți piticii să-i viziteze pe toți ceilalți într-o perioadă scurtă de timp.

O dată, oamenii care locuiau în țările vecine *Țării Piticilor* au hotărât să construiască unele autostrăzi în linii drepte. Deoarece oamenii nu aveau nici un fel de rețineri față de pitici, ei au proiectat anumite autostrăzi astfel încât acestea urmau să traverseze *Țara Piticilor*. Piticii au descoperit aceste planuri și au fost extrem de nefericiți, deoarece, ei fiind mici și lenți, nu puteau traversa aceste autostrăzi fără ca viața lor să nu fie în pericol.

Piticii au făcut rost de proiectele realizate de oameni și acum au nevoie de ajutorul vostru. Ei doresc să se viziteze și în viitor, deci nu le convin deloc acele autostrăzi care împart mulțimea caselor lor în două părți nevide. După ce vor afla care sunt acele autostrăzi care lor nu le plac, ei vor împiedica pe oameni prin vrăjile lor să le construiască.

Se dă numărul N, reprezentând numărul punctelor (caselor) din plan și mai multe linii drepte (autostrăzi). Determinați, corespunzător fiecărei linii, dacă cele N puncte se situează în același semiplan determinat de linia dată, sau nu. Pentru fiecare linie prelucrată va trebui să afișați răs-

punsul imediat după prelucrare, adică înainte de citirea datelor referitoare la linia următoare. Se garantează că nici o autostradă nu trece printr-o casă.

Date de intrare și ieșire

Programul va trebui să citească datele de la *intrarea standard* (`stdin` în *C/C++*, `input` în *FreePascal*) și să scrie datele de ieșire la *ieșirea standard* (`stdout` în *C/C++*, `output` în *FreePascal*).

Prima linie a intrării conține un număr întreg N ($0 \le N \le 100000$). Pe fiecare linie i dintre următoarele N se află două numere reale x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$), separate printr-un singur spațiu, care reprezintă coordonatele celei de a i-a case.

Următoarele linii conțin, fiecare patru numere reale X_1, Y_1, X_2, Y_2 ($-10^9 \le X_1, Y_1, X_2, Y_2 \le 10^9$) separate prin câte un spațiu. Aceste numere reprezintă coordonatele (X_1, Y_1) și (X_2, Y_2) a două puncte diferite, aflate pe autostradă. Corespunzător fiecărei linii de la intrare, programul vostru va trebui să afișeze pe o linie mesajul `GOOD`, dacă toate punctele date se află în același semiplan determinat de linia dată, sau `BAD`, dacă există puncte de o parte și alta a liniei. După afișarea unui răspuns la ieșirea standard va trebui să realizați operația *flush* asupra *buffer*-ului de ieșire, așa cum se va arăta în continuare.

Sistemul de evaluare va opri execuția programelor voastre după ce acesta a furnizat răspunsul aferent ultimei autostrăzi citite. Programele voastre nu trebuie să își încheie singure execuția. Se garantează că există cel mult 100000 de autostrăzi.

Instrucțiuni pentru programatorii în C/C++

Pentru citirea unei linii de la intrarea standard va trebui să folosiți o secvență de tipul:

```
double X_1,Y_1,X_2,Y_2;
scanf(" %lf %lf %lf %lf",&X_1,&Y_1,&X_2,&Y_2);
```

Observați că nu există spațiu după ultimul `%lf`.

Afișarea ieșirii corespunzătoare unei singure linii se realizează astfel:

```
printf("GOOD\n");
fflush(stdout);
```

Instrucțiuni pentru programatorii în Pascal

Pentru citirea unei linii de la intrarea standard va trebui să folosiți o secvență de tipul:

```
var X_1,Y_1,X_2,Y_2:Double;
Read(X_1,Y_1,X_2,Y_2);
```

Afișarea ieșirii corespunzătoare unei singure linii se realizează astfel:

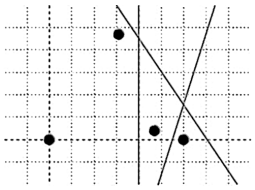
```
Writeln('GOOD');
Flush(Output);
```

Atenție!!!

Sunteți sfătuiți să folosiți tipul de date `double` (atât în *C/C++*, cât și *FreePascal*) pentru a reține numerele reale. Rețineți că în aritmetica numerelor reale pot să apară erori de rotunjire. Egalitatea a două numere reale x și y o veți verifica nu prin a testa dacă $x = y$, ci prin a testa dacă este satisfăcută relația $|x - y| < \varepsilon$ (unde ε este o constantă mai mică sau egală cu 10^{-4}).

Exemplu

Intrarea standard	*Ieșirea standard*
4	GOOD
0.0 0.0	BAD
6.00 -0.001	BAD
3.125 4.747	
4.747 0.47	
5 3 7 0	
4 -4.7 7 4.7	
4 47 4 94	



Timp maxim de execuție/test: 5 secunde
Memorie disponibilă: 16 MB

P060211: Gărzile regale

A fost o dată ca niciodată un regat care avea un rege cam paranoic. Castelul său avea fundația de formă dreptunghiulară care fusese împărțită în $M \times N$ pătrate având latura egală cu unitatea. Unele pătrate corespundeau unor ziduri, altele erau libere și corespundeau unor camere. Din cauza paranoii, regele a hotărât ca în anumite camere să amplaseze capcane ascunse.

Dar tot nu era mulțumit. A decis să așeze în castel cât mai mulți paznici posibil. Dar acest fapt nu era un lucru simplu de rezolvat. Paznicii au fost antrenați astfel încât să împuște orice persoană imediat după ce au zărit-o. Rezultă că regele trebuia să-și amplaseze paznicii cu grijă, deoarece dacă doi paznici se zăreau unul pe celălalt, ei se împușcau între ei. Evident, regele nu putea să amplaseze paznici în camerele cu capcane.

Din cauză că doi paznici aflați într-o aceeași cameră se vedeau, rezultă că într-o cameră se putea afla un singur paznic. Doi paznici aflați în camere diferite se zăreau dacă și numai dacă pătratele corespunzătoare camerelor în care se aflau erau pe aceeași linie sau pe aceeași coloană a planului castelului și dacă nu exista zid între ei.

Un paznic poate supraveghea patru direcții conform modului de deplasare a turei pe tabla de șah.

Determinați numărul maxim de paznici care se pot amplasa în interiorul castelului (conform regulilor de mai sus) și dați o posibilă amplasare a lor în camerele castelului.

Date de intrare

Pe prima linie a fișierului de intrare **`GUARDS.IN`** se află două numere M și N ($1 \le M, N \le 200$), reprezentând dimensiunile planului fundației castelului. Fiecare a i-a linie dintre următoarele M conține N numere $a_{i,1}, ..., a_{i,N}$, separate printr-un singur spațiu, unde:

- $a_{i,j} = 0$ indică faptul că pătratul de coordonate (i, j) este liber (corespunde unei camere fără capcană);

- $a_{i,j}$ = 1 indică faptul că pătratul de coordonate (i, j) conține o capcană;
- $a_{i,j}$ = 2 indică faptul că pătratul de coordonate (i, j) reprezintă un zid.

Prima coordonată a pătratului reprezintă linia, iar cea de a doua reprezintă coloana.

Date de ieșire

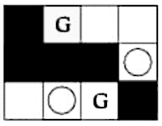
Prima linie a fișierului de ieșire **GUARDS.OUT** va trebui să conțină cel mai mare număr K de paznici care pot fi amplasați în interiorul castelului. Pe următoarele K linii va trebui să se afișeze o amplasare posibilă a celor K paznici în camere libere astfel încât doi paznici să nu se poată zări unul pe altul.

Mai exact, cea de a i-a linie dintre cele K va trebui să conțină două numere întregi r_i și c_i, separate printr-un singur spațiu, care reprezintă coordonatele camerei în care se află al i-lea paznic (linia este specificată prin r_i, iar coloana prin c_i).

Exemplu

GUARDS.IN
```
3 4
2 0 0 0
2 2 2 1
0 1 0 2
```

GUARDS.OUT
```
2
1 2
3 3
```



Timp maxim de execuție/test: 15 secunde
Memorie disponibilă: 8 MB

P060212: Petrecere aniversară

Se apropie ziua de naștere a lui *John* și, ca în fiecare an, el dorește să organizeze un mare chef. Ar dori ca toți prietenii săi să vină la petrecere, dar constată supărat că acest lucru este aproape imposibil. De exemplu, *Susie* l-a părăsit pe *Steve* săptămâna trecută, deci este aproape imposibil să-i aibă pe amândoi alături la chef.

John a pierdut mult timp vizitând prietenii și rugându-i să vină la petrecere. A primit o serie de promisiuni, dar și mai multe rugăminți.

"*Dacă mă inviți pe mine, trebuie să-l inviți și pe prietenul meu*!" a precizat *Veronica*.

"*Dacă inviți gemenii Burdiliak, nu te aștepta ca eu sau Joseph să venim*!" l-a avertizat *Peter*.

Brusc, *John* și-a dat seama, că va fi foarte greu să satisfacă toate solicitările primite.

Veți primi o descriere a solicitărilor pe care *John* le-a primit de la prietenii săi. Va trebui să determinați un grup de persoane astfel încât, dacă *John* invită persoanele aparținând acestui grup (și pe nimeni altul), toate solicitările adunate să fie satisfăcute.

Solicitările sunt descrise în felul următor:

- `name` este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă *John* îl invită pe `name` (`name` este un șir care conține cel mult 20 de litere mici ale alfabetului englez);
- `-name` este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă *John* nu îl invită pe `name` (`name` este un șir care conține cel mult 20 de litere mici ale alfabetului englez);
- dacă `R1`, ..., `Rk` sunt solicitări, atunci (`R1 & ... & Rk`) este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă toate solicitările `R1`, ..., `Rk` sunt satisfăcute;
- dacă `R1`, ..., `Rk` sunt solicitări, atunci (`R1 | ... | Rk`) este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă cel puțin una dintre solicitările `R1`, ..., `Rk` este satisfăcută;
- dacă `R1` și `R2` sunt solicitări, atunci `(R1 => R2)` este o solicitare. Această solicitare ***nu*** va fi satisfăcută dacă și numai dacă solicitarea `R1` este satisfăcută și solicitarea `R2` nu este satisfăcută.

Date de intrare

Concurenții au primit zece fișiere de intrare, denumite **PARTY1.IN**, **PARTY2.IN**, ..., **PARTY10.IN**. Fiecare dintre acestea conține un set de date de intrare.

Pe prima linie a unui fișier de intrare se află numărul F al prietenilor lui *John*, pe următoarele F linii se află numele prietenilor, câte unul pe o linie.

Pe următoarea linie se află numărul N al solicitărilor. Următoarele N linii conțin fiecare câte o solicitare de forma celor cinci specificate în enunț.

Date de ieșire

Concurenții au furnizat zece fișiere de ieșire, denumite **PARTY1.OUT**, **PARTY2.OUT**, ..., **PARTY10.OUT**. Prima linie a unui fișier de ieșire corespunzător unui set de date (fișier de intrare) ar trebui să conțină numărul K al persoanelor pe care le va invita *John*. Următoarele K linii ar trebui să conțină numele acestora, câte unul pe linie. Se garantează existența unei soluții (nu neapărat unice) pentru fiecare fișier de intrare. Dacă există mai multe soluții, trebuie furnizată doar una dintre ele.

Exemplu

PARTY0.IN
```
3
veronica
steve
dick
3
(veronica => dick)
(steve => -veronica)
(steve & dick)
```

PARTY0.OUT

Observație

Pentru fiecare dintre cele zece fișiere de intrare, trebuia construit câte un fișier de ieșire. La sfârșitul concursului, concurenții au trimis aceste fișiere, la fel cum au trimis codurile sursă pentru celelalte probleme. Ei nu erau obligați să scrie un program dacă reușeau să genereze fișierele de ieșire corecte prin alte metode.