PROBLEME propuse pentru REZOLVARE

Vă prezentăm enunțurile celor șase probleme propuse spre rezolvare concurenților de la Olimpiada Internațională de Informatică, desfășurată în perioada 14-21 iulie 2001 în orașul finlandez Tampere.

P060101: Telefoane mobile

În *Tampere* stațiile de telefonie mobilă din generația a patra funcționează după cum urmează.

Teritoriul orașului este împărțit în pătrate. Pătratele formează o matrice de dimensiunea $S \times S$, liniile și coloanele căreia sunt indexate de la 0 la S - 1. Fiecare pătrat conține câte o stație de telefonie. Întrucât telefoanele pot fi mutate dintr-o zonă în alta, pot fi pornite sau oprite, numărul de telefoane mobile active din cadrul oricărui pătrat se poate schimba. La cererea stației principale, stațiile din interiorul pătratelor indicate îi transmit, fiecare, numărul curent de telefoane active.

Scrieți un program care recepționează datele transmise de stații și furnizează răspunsuri referitoare la numărul de telefoane mobile active din zonele dreptunghiulare indicate.

Intrări și ieșiri

Datele de intrare sunt citite sub formă de numere întregi de la intrarea standard, iar răspunsurile sunt scrise, de asemenea, ca numere întregi la ieșirea standard.

Datele de intrare sunt codificate după cum urmează. Fiecare linie a fișierului de intrare conține o instrucțiune formată dintr-un cod - un număr întreg (N) - eventual urmat de parametri, care de asemenea sunt numere întregi.

Valorile posibile ale parametrului N sunt 0, 1, 2 și 3. Semnificațiile acestora sunt descrise în continuare.

În cazul în care valoarea lui N este 0, atunci instrucțiunea are un singur parametru (S) și indică inițializarea unei matrice de dimensiunea $S \times S$ ale cărei elemente vor avea toate valoarea 0. Această instrucțiune apare numai o singură dată și întotdeauna este prima în fișierul de in fișierul de intrare.

Dacă valoarea lui N este 1, instrucțiunea are trei parametri (X, Y și A) și indică faptul că numărului de telefoane active din pătratul având coordonatele (X, Y) i se adaugă valoarea A, care poate fi pozitivă sau negativă.

Pentru valoarea 2 a lui N, vom avea patru parametri (L, B, R și T). Instrucțiunea înscrie în fișierul de ieșire suma numerelor de telefoane active din toate pătratele având coordonatele (X, Y), unde $L \le X \le R$, $B \le Y \le T$.

Valoarea 3 a lui N indică terminarea programului. Această instrucțiune apare numai o singură dată și întotdeauna este ultima în fișierul de intrare.

Valorile de la intrare vor fi întotdeauna din intervalul indicat în restricții, deci nu se cere verificarea lor. La fel, orice valoare negativă a numărului A nu va putea conduce la o valoare negativă în matrice. Indexarea pătratelor începe de la 0, prin urmare pentru o matrice de dimensiunea 4×4 avem $0 \le X \le 3$ și $0 \le Y \le 3$.

Programul dumneavoastră nu va înscrie nimic în fișierul de ieșire în cazul instrucțiunilor ce diferă de instrucțiunea având codul 2. Dacă instrucțiunea curentă din fișierul de intrare are codul 2, se cere ca programul să scrie pe câte o linie în fișierul standard de ieșire numărul calculat de telefoane active.

Instrucțiuni de programare

În exemplele ce urmează variabila întreagă `last` conține numărul curent citit din fișierul de intrare, iar variabila întreagă `answer` conține numărul de telefoane calculat de program.

În cazul programelor scrise în C++ și utilizarea *iostreams*, se va folosi următoarea secvență pentru citirea de la intrarea standard și scrierea la ieșirea standard:

```
cin>>last;
cout<<answer<<endl<<flush;
```

În cazul programelor scrise în C sau C++ și utilizarea funcțiilor `scanf()` și `printf()`, se va folosi următoarea secvență pentru citirea de la intrarea standard și scrierea la ieșirea standard:

```
scanf("%d",&last);
printf("%d\n",answer); fflush(stdout);
```

În cazul programelor scrise în *Pascal* se va folosi următoarea secvență pentru citirea de la intrarea standard și scrierea la ieșirea standard:

```
Read(last); ... Readln;
Writeln(answer);
```

Exemplu

Linia	**stdin**	**stdout**
1.	0 4	
2.	1 1 2 3	
3.	2 0 0 2 2	
4.		3
5.	1 2	
6.	1 1 2 -1	
7.	2 1 1 2 3	
8.		4
9.	3	

Explicații

- prima linie indică inițializarea unei matrice de dimensiunea 4 × 4;
- a doua linie adaugă valoarea +3 la elementul având coordonatele (1, 2) din matrice;
- a treia linie calculează suma elementelor din matrice cu indicii (X, Y), unde $0 \le X \le 2$, $0 \le Y \le 2$;
- a patra linie specifică faptul că la ieșirea standard se va afișa valoarea 3;
- a cincia linie adugă valoarea +2 la elementul având coordonatele (1, 1) din matrice;
- a șasea linie adaugă valoarea -1 la elementul având coordonatele (1, 2) din matrice;
- a șaptea linie calculează suma elementelor din matrice având indicii (X, Y), unde $1 \le X \le 2$, $1 \le Y \le 3$;
- a opta linie specifică faptul că la ieșirea standard se va afișa valoarea 4;
- a noua linie indică faptul că programul se încheie.

Restricții

- Matricea pătratică va avea cel mult 1024 de linii și cel mult 1024 de coloane: $1 \times 1 \le S \times S \le 1024 \times 1024$.
- În fiecare moment valoarea V corespunzătore unui pătrat va fi un număr întreg pozitiv cu semn care poate fi reprezentat pe 16 biți: $0 \le V \le 2^{15} - 1$ (= 32767).
- Valoarea A va fi un număr întreg cu semn care poate fi reprezentat pe 16 biți; $-2^{15} \le A \le 2^{15} - 1$ (= 32767).
- Numărul U al instrucțiunilor de la intrare va fi cuprins între 3 și 60002: $3 \le U \le 60002$.
- Suma elementelor matricei este cel mult M ($=2^{30}$).

Precizări

- Timpul maxim de execuție admis este de 1 secundă/test.
- Limita maximă a memoriei care poate fi folosită este de 5 MB.
- Au fost 20 de teste, pentru rezolvarea fiecăruia acordându-se cel mult 5 puncte. Pentru 16 dintre ele dimensiunea matricei era de cel mult 512 × 512.
- Opțiunile de compilare folosite pentru limbajul *Pascal* au fost -So -O2 -Xs; opțiunile de compilare folosite pentru limbajele *C* și *C++* au fost -O2 -static;
- Facilitățile de testare în rețea au permis ca fișierele de test să fie redirectate la intrarea standard a programului concurentului.

P060102: Jocul Ioiwari

Familia de jocuri *Mancala* cu mărgele și adâncituri într-o tablă de joc este una dintre cele mai vechi forme de distracție. Această problemă prezintă o versiune a jocului, concepută special pentru IOI.

La joc participă doi jucători care folosesc o tablă rotundă, prevăzută cu șapte adâncituri. În plus, fiecare jucător are propria sa bancă de mărgele. Jocul începe prin distribuirea aleatoare a 20 de mărgele în adâncituri; fiecare adâncitură conține cel puțin două și cel mult patru mărgele.

Cei doi jucători mută alternativ. La fiecare mutare, jucătorul alege o adâncitură nevidă, ia toate mărgelele din ea și le păstrează în mână. Apoi, atât timp cât jucătorul mai are mărgele în mână, analizează pe rând următoarele adâncituri, în sensul acelor de ceasornic, începând cu cea care urmează după cea pe care a golit-o. Pentru fiecare dintre aceste adâncituri execută una dintre următoarele operații:

- Dacă are mai mult de o mărgea în mână:
  - ♦ dacă în adâncitura curentă există deja cinci mărgele, atunci ia una dintre ele și o pune în banca sa;
  - ♦ în caz contrar pune în adâncitură una din mărgelele pe care le ține în mână.

- Dacă are exact o mărgea în mână:
  - ♦ dacă adâncitura curentă conține cel puțin una și cel mult patru mărgele, atunci jucătorul mută în bancă toate mărgelele din adâncitură, precum și pe cea din mâna sa;
  - ♦ în caz contrar (adâncitura conține 0 sau 5 mărgele), jucătorul plasează mărgeaua din mâna sa în banca jucătorului advers.

Jocul se termină când toate adâncițurile sunt goale. În această situație, câștigătorul este acel jucător care are în bancă cele mai multe mărgele.

Jucătorul care începe jocul are întotdeauna strategie sigură de câștig. Scrieți un program care joacă *Ioiwari* ca prim jucător și câștigă. Adversarul joacă optim, adică o dată ce i se dă ocazia, profită de ea și câștigă jocul.

Intrări și ieșiri

Programul citește datele de intrare de la intrarea standard și scrie la ieșirea standard. Programul este Jucătorul 1, iar ad-

versarul este Jucătorul 2. Programul începe prin citirea de pe prima linie a șapte numere întregi (p_1, ..., p_7) care reprezintă numărul inițial de mărgele din cele șapte adâncituri, în această ordine. Adâncimile sunt etichetate cu numere întregi de la 1 la 7 în sensul acelor de ceasornic. După aceasta, jocul începe cu băncile fără mărgele. Programul trebuie să joace în modul următor:

- dacă este rândul programului să mute, atunci programul va scrie la ieșirea standard eticheta adâncimii pe care o alege ca prim pas al mutării;
- dacă este rândul adversarului să mute, atunci programul vostru va citi de la intrarea standard eticheta adâncimii pe care adversarul o alege ca prim pas al mutării.

Instrumente

Programul jucătorului advers (`ioiwari2` în *Linux*, și `ioiwari2.exe` în *Windows*) joacă, plecând de la o poziție inițială dată, optim ca Jucătorul 2. Acesta va scrie mai întîi la ieșirea standard o primă linie, care va fi citită de programul vostru; această linie descrie numărul inițial de mărgele din adâncituri pentru jocul respectiv. O posibilitate ar fi: `4 3 2 4 2 3 2`.

Apoi, programul va juca încercând să citească mutările Jucătorului 1 de la intrarea standard și scriind propriile mutări la ieșirea standard.

Puteți rula programul vostru și programul `ioiwari2` în ferestre separate și puteți transfera manual conversația între programe. Conversația este memorată în fișierul `ioiwari.out`.

Instrucțiuni de programare

În exemplele următoare, variabila `last` conține cea mai recentă mutare a adversarului, mutarea voastră va apărea în variabila `mymove`.

Dacă programați în *C++* și folosiți *iostreams*, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard exact după cum urmează:

```
cout<<mymove<<endl<<flush;
cin>>last;
```

Dacă programați în *C* sau *C++* și folosiți funcțiile `scanf()` și `printf()`, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard exact după cum urmează:

```
printf("%d\n",mymove); fflush(stdout);
scanf("%d",&last);
```

Dacă programați în *Pascal*, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard exact după cum urmează:

```
Writeln(mymove);
Readln(last);
```

Exemplu

Fie următoarea secvență corectă de șase mutări:

Operație		*Etichetă adâncitură*								
Juc.	*Mutare*	1	2	3	4	5	6	7	*Banca*1	*Banca*2
situația inițială:		4	3	2	4	2	3	2	0	0
1.	2	4	0	3	5	0	3	2	3	0
2.	3	4	0	0	4	1	4	0	3	4
1.	5	4	0	0	4	0	0	0	8	4
2.	4	0	0	0	0	1	1	1	8	9
1.	5	0	0	0	0	0	0	1	10	9
2.	7	0	0	0	0	0	0	0	11	9

Mutările efectuate, conținutul adâncimilor și al băncilor

Modul de punctare

Pentru fiecare test primiți patru puncte dacă programul vostru câștigă jocul, două puncte în caz de remiză, respectiv zero puncte dacă programul vostru pierde jocul.

Precizări

- Timpul maxim de execuție admis este de 1 secundă/test. Programul care reprezintă adversarul este executat ca un proces separat al cărui timp de execuție nu este adunat la timpul de execuție al programului vostru.
- Limita maximă a memoriei care poate fi folosită este de 32 MB.
- Au fost 25 de teste, pentru rezolvarea fiecăruia acordându-se cel mult 4 puncte.
- Opțiunile de compilare folosite pentru limbajul *Pascal* au fost `-So -O2 -XS`; opțiunile de compilare folosite pentru limbajele *C* și *C++* au fost `-O2 -static`.

P060103: Doi-cinci

Mesajele secrete dintre *Moș Crăciun* și micile sale ajutoare sunt, de obicei, codificate în limbajul 25. Alfabetul limbajului 25 este același cu alfabetul latin, cu o excepție - litera `'Z'` este absentă, adică alfabetul conține cele 25 de litere ale alfabetului latin. Fiecare cuvânt al limbajului 25 este format din exact 25 de litere distincte.

Un cuvânt poate fi scris într-un tablou cu dimensiunea 5 × 5, completând mai întâi liniile; de exemplu, cuvântul `ADJPTBEKQUCGLRVFINSWHMOXY` va fi scris după cum urmează:

```
A   D   J   P   T
B   E   K   Q   U
C   G   L   R   V
F   I   N   S   W
H   M   O   X   Y
```

Un cuvânt corect în limbajul 25 are literele pe fiecare linie și pe fiecare coloană în ordine crescătoare. Astfel, cuvântul `ADJPTBEKQUCGLRVFINSWHMOXY` este un cuvânt corect, spre deosebire de `ADJPTBEGQUCKLRVFINSWHMOXY`, care nu este un cuvânt corect (ordinea crescătoare este încălcată în coloanele a doua și a treia).

Moș Crăciun are un lexicon. Lexiconul conține lista cu toate cuvintele corecte din limbajului 25, în ordine lexicografică crescătoare, însoțite de numerele lor de ordine, începând de la 1.

De exemplu, cuvântul `ABCDEFGHIJKLMNOPQRSTUVWXY` este primul în lexicon, iar `ABCDEFGHIJKLMNOPQRSUTVWXY`, este al doilea cuvânt. Se observă că singura diferență dintre cele două cuvinte este interschimbarea literelor `U` și `T`.

Din nefericire, acest lexicon este uriaș. Scrieți un program care determină numărul de ordine al unui cuvânt dat precum și cuvântul corespunzător unui număr de ordine dat. Există cel mult 2^{31} cuvinte în lexicon.

Intrarea

Numele fișierului de intrare este `TWOFIVE.IN` și conține două linii. Pe prima linie apare una dintre literele `'W'` sau `'N'`.

- Dacă pe prima linie apare `'W'`, atunci a doua linie conține un cuvânt corect din limbajul 25.
- Dacă pe prima linie apare `'N'`, atunci a doua linie conține numărul de ordine al unui cuvânt corect existent în limbajului 25.

Ieșirea

Numele fișierului de ieșire este `twofive.out` și conține o singură linie.

- Dacă pe a doua linie a fișierului de intrare apare un cuvânt, atunci linia fișierului de ieșire trebuie să conțină numărul de ordine al acestui cuvânt.
- Dacă pe a doua linie a fișierului de intrare apare un număr, atunci linia fișierului de ieșire trebuie să conțină cuvântul care are acel număr de ordine.

Exemple

`twofive.in`
```
ABCDEFGHIJKLMNOPQRSUTVWXY
```

`twofive.out`
```
2
```

`twofive.in`
```
2
```

`twofive.out`
```
ABCDEFGHIJKLMNOPQRSUTVWXY
```

Precizări

- Timpul maxim de execuție admis este de 0.02 secunde/test.
- Limita maximă a memoriei care poate fi folosită este de 32 MB.
- Au fost 20 de teste, pentru rezolvarea fiecăruia acordându-se cel mult 5 puncte.
- Opțiunile de compilare folosite pentru limbajul *Pascal* au fost `-So -O2 -XS`; opțiunile de compilare folosite pentru limbajele *C* și *C++* au fost `-O2 -static`.

P060104: Score

Score este un joc pentru doi jucători care mută un jeton dintr-o poziție în alta a unei table. Tabla are N poziții, numerotate de la 1 la N și un număr de săgeți. Fiecare săgeată pleacă dintr-o poziție către o altă poziție. Fiecare poziție este proprietatea unui singur jucător, numit proprietarul poziției respective. În plus, fiecărei poziții i se asociază o valoare întreagă pozitivă. Valorile asociate pozițiilor sunt distincte. Poziția 1 este poziția de start. Inițial, ambii jucători au scorul 0.

Jocul se desfășoară în modul următor. Notăm cu C poziția curentă a jetonului la începutul mutării. La începutul jocului, poziția curentă C este poziția 1. O mutare constă din următoarele operații:

- Dacă valoarea lui C este mai mare decât scorul curent al proprietarului poziției C, atunci valoarea acestei poziții devine noul scor al proprietarului poziției C; în caz contrar, scorul proprietarului poziției C nu se modifică. Scorul celuilalt jucător nu se modifică în nici unul dintre cazurile de mai sus.
- Proprietarul poziției C alege una dintre săgețile care pleacă din poziția curentă a jetonului și mută jetonul în poziția în care se află vârful săgeții.

Se observă faptul că un anumit jucător poate efectua mai multe mutări consecutive.

Jocul se termină atunci când jetonul revine în poziția de start. Câștigă jucătorul care la terminarea jocului are cel mai mare scor.

Săgețile sunt întotdeauna aranjate astfel încât:

- întotdeauna din poziția curentă a jetonului pleacă cel puțin o săgeată;
- fiecare poziție P este accesibilă din poziția de start, adică există o succesiune de săgeți de la poziția de start la poziția P;
- se garantează că jocul se termină după un număr finit de mutări.

Scrieți un program care să joace acest joc și să îl câștige. Toate jocurile propuse în evaluare vă permit să câștigați, indiferent dacă începeți sau nu primul. Jucătorul advers din evaluare joacă optimal, în sensul că dacă i se oferă o șansă de câștig, atunci va profita de ea și va câștiga jocul.

Intrări și ieșiri

Programul vostru citește de la intrarea standard și scrie la ieșirea standard. Programul vostru este Jucătorul 1, iar adversarul este Jucătorul 2. La început, vor trebui citite următoarele date de la intrarea standard:

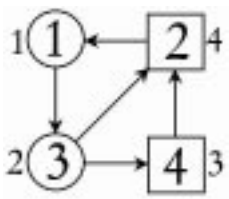
- prima linie conține un număr întreg N care reprezintă numărul pozițiilor ($1 \le N \le 1000$);
- fiecare dintre următoarele N linii conțin câte N numere întregi care descriu săgețile astfel: dacă există o săgeată de la poziția i la poziția j, atunci cel de-al j-lea număr de pe a i-a linie este 1, iar dacă o astfel de săgeată nu există, atunci numărul respectiv va fi 0;

- următoarea linie conține N numere întregi care indică proprietarii pozițiilor; dacă poziția i îl are ca proprietar pe Jucătorul 1 (programul vostru), atunci cel de-al i-lea întreg este 1, iar în caz contrar este 2.
- următoarea linie conține N numere întregi care indică valorile pozițiilor; dacă cel de-al i-lea întreg este j, atunci valoarea asociată poziției i este j; toate valorile j satisfac relația $1 \le j \le N$ și sunt distincte.

După citirea acestor date jocul va începe, iar jetonul se află în poziția 1.

Programul se va încheia în momentul în care jetonul revine în poziția 1 și se desfășoară astfel:
- Dacă este rândul programului vostru să mute, atunci el va scrie la ieșirea standard numărul P corespunzător poziției următoare ($1 \le P \le N$).
- Dacă este rândul adversarului să mute, atunci programul vostru trebuie să citească de la intrarea standard numărul P corespunzător poziției următoare ($1 \le P \le N$).

Să considerăm tabla din figură. Pozițiile încadrate într-un cerc aparțin Jucătorului 1, iar cele încadrate într-un pătrat aparțin Jucătorului 2.



Valoarea fiecărei poziții este scrisă în interiorul cercului sau pătratului corespunzător, iar numărul poziției este precizat lângă pătrat sau cerc. De exemplu, un joc se desfășoară după cum urmează:

Exemplu

Linia	**stdin**	**stdout**
1.	4	
2.	0 1 0 0	
3.	0 0 1 1	
4.	0 0 0 1	
5.	1 0 0 0	
6.	1 1 2 2	
7.	1 3 4 2	
8.		2
9.		4
10.	1	

Explicații
- prima linie a intrării indică faptul că tabla are patru poziții;
- a doua linie indică faptul că din prima poziție pleacă o singură săgeată care are vârful în a doua poziție;
- a treia linie arată că din a doua poziție pleacă două săgeți care au vârfurile în a doua și a treia poziție;
- conform celei de-a patra linii, din a treia poziție pleacă o singură săgeată care are vârful în a patra poziție;
- cea de-a cincea linie arată că din a patra poziție pleacă tot o săgeată care are vârful în prima poziție;
- proprietarii celor patru poziții sunt precizați în cea de-a șasea linie: primul jucător deține prima și a doua poziție, iar al doilea deține a treia și a patra poziție;
- valorile celor patru poziții sunt indicate în cea de-a șaptea linie;
- prima linie a ieșirii (linia a opta) indică faptul că primul jucător mută jetonul în a doua poziție;
- cea de-a doua linie a ieșirii (linia a noua) indică faptul că primul jucător mută jetonul în a patra poziție;
- ultima linie a intrării (linia a zecea) indică faptul că al doilea jucător a mutat jetonul în prima poziție și jocul se încheie.

La sfârșitul jocului, Jucătorul 1 are scorul 3, iar Jucătorul 2 are scorul 2, deci Jucătorul 1 câștigă.

Instrucțiuni de programare
În exemplele următoare `target` este o variabilă care memorează poziția curentă.

Dacă programați în C++ și utilizați *iostreams*, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard astfel:

```
cin>>target;
cout<<target<<endl<<flush;
```

Dacă programați în *C* sau *C++* și utilizați funcțiile `scanf()` și `printf()`, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard astfel:

```
scanf("%d",&target);
printf("%d\n",target); fflush(stdout);
```

Dacă programați în *Pascal*, trebuie să implementați citirea de la intrarea standard și scrierea la ieșirea standard astfel:

```
Writeln(target);
Readln(target);
```

Instrumente
Aveți la dispoziție un program (`score2` pentru sistemul de operare *Linux* și `score2.exe` pentru sistemul de operare *Windows*).

Programul citește din fișierul `score.in` descrierea jocului în formatul prezentat anterior și scrie această informație la ieșirea standard folosind același format. Această ieșire poate fi folosită de programul vostru ca intrare pentru testare.

În continuare programul joacă folosind o strategie aleatoare citind mutările programului vostru de la intrarea standard și scriind propriile lui mutări la ieșirea standard.

Punctaj și evaluare
Pentru fiecare test la care programul vostru câștigă jocul, veți primi punctele corespunzătoare. Dacă jocul este pierdut nu veți primi nici un punct. În timpul evaluării progra-

mul vostru va juca împotriva unui alt program având la dispoziție o limită de timp cu o secundă mai mare decât cea de la evaluarea propriu-zisă. Intrările și ieșirile programului sunt păstrate, iar apoi programul este executat pentru a doua oară, dar intrarea va fi redirectată spre un fișier. În acest moment va fi măsurat timpul oficial de execuție. La această a doua execuție programul trebuie să furnizeze aceleași ieșiri pe care le-a furnizat la prima execuție.

Precizări

- Timpul maxim de execuție admis este de 1 secundă/test.
- Limita maximă a memoriei care poate fi folosită este de 32 MB.
- Au fost 20 de teste, pentru rezolvarea fiecăruia acordându-se cel mult 5 puncte.
- Opțiunile de compilare folosite pentru limbajul *Pascal* au fost `-So -O2 -XS`; opțiunile de compilare folosite pentru limbajele *C* și *C++* au fost `-O2 -static`.

P060105: Criptare dublă

Standardul avansat de criptare (***AES*** - ***A**dvanced **E**ncryption **S**tandard*) utilizează un algoritm nou și puternic. Acest algoritm operează cu trei blocuri de câte 128 de biți. Pentru un bloc necriptat p (textul inițial) și un bloc cheie k, funcția de criptare E returnează un bloc criptat c (textul criptat):

$$c = E(p, k).$$

Funcția de decriptare D este inversa funcției de criptare E:

$$D(E(p, k), k) = p,$$
$$E(D(c, k), k) = c.$$

În standardul *Dublu AES*, două blocuri cheie independente k_1 și k_2 sunt utilizate consecutiv, mai întâi k_1 și apoi k_2:

$$c_2 = E(E(p, k_1), k_2).$$

În această problemă se furnizează și un număr întreg s. Doar cei $4 \cdot s$ biți din stânga ai cheilor sunt relevanți, ceilalți biți (cei $128 - 4 \cdot s$ din dreapta cheii) având toți valoarea 0.

Se cere să reconstituiți perechea de chei de criptare pentru unele blocuri criptate cu *Dublu AES*. Se cunosc blocul inițial p, blocul criptat c_2 și numărul întreg s cu semnificația de mai sus.

Algoritmul *AES* de criptare și algoritmul *AES* de decriptare sunt disponibili într-o bibliotecă.

Nu va trebui să realizați un program care rezolvă problema, ci doar să descoperiți perechile de chei corespunzătoare seturilor de date.

Intrare

Se dau zece instanțe ale problemei și anume fișierele text `double1.in`, `double2.in`, ..., `double10.in`. Fiecare fișier este format din trei linii. Prima linie conține numărul întreg s, linia a doua conține blocul necriptat p, iar linia a treia conține blocul criptat c_2 obținut din p cu ajutorul criptării *Dublu AES*. Ambele blocuri sunt citite ca șiruri de caractere ce conțin 32 de cifre hexazecimale (`'0'..'9'`, `'A'..'F'`). Biblioteca care va fi prezentată în continuare vă pune la dispoziție o rutină pentru transformarea șirurilor de caractere în blocuri. Toate fișierele de intrare admit soluții.

Ieșirea

Trebuie să creați zece fișiere de ieșire, corespunzătoare fișierelor de intrare date. Fiecare fișier de ieșire conține trei linii: pe prima linie se va afla textul `#FILE double I`, unde `I` este numărul fișierului de intrare, pe a doua linie se află cheia k_1, iar linia a treia cheia k_2, astfel încât

$$c_2 = E(E(p, k_1), k_2).$$

Ambele blocuri trebuie scrise ca șiruri de caractere ce conțin 32 de cifre hexazecimale (`'0'..'9'`, `'A'..'F'`). Biblioteca vă pune la dispoziție o rutină pentru transformarea blocurilor în șiruri de caractere. Dacă există mai multe soluții, atunci va fi generată una singură.

Exemplu

Pentru exemplificare, vom folosi fișierul de intrare cu numărul `0`.

double0.in

```
1
00112233445566778899AABBCCDDEEFF
6323B4A5BC16C479ED6D94F5B58FF0C2
```

Un posibil conținut al fișierului de ieșire ar putea fi:

```
#FILE double 0
A0000000000000000000000000000000
7000000000000000000000000000000
```

Biblioteca

Partea declarativă a bibliotecii pentru limbajul *FreePascal* este următoarea:

```
type HexStr=string[32]; { '0'..'9', 'A'..'F' }
     Block=array[0..15] of Byte; { 128 bits }

procedure HexStrToBlock(const hs:HexStr;
                                  var b:Block);

procedure BlockToHexStr(const b:Block;
                                 var hs:HexStr);

procedure Encrypt(const p,k:Block;
                                  var c:Block);
  { c = E(p,k) }

procedure Decrypt(const c,k:Block;
                                  var p:Block);
  { p = D(c,k) }
```

Programul `aestoolp.pas` ilustrează modul de utilizare a bibliotecii *FreePascal*.

Partea declarativă a bibliotecii pentru *GNU C/C++* este următoarea:

```
typedef char HexStr[33];
       // '0'..'9', 'A'..'F', '\0'-terminator

typedef unsigned char Block[16];  // 128 bits

void hexstr2block(const HexStr hs, Block b);

void block2hexstr(const Block b, HexStr hs);

void encrypt(const Block p, const Block k,
                                    Block c);
  // c = E(p,k)

void decrypt(const Block c, const Block k,
                                    Block p);
  // p = D(c,k)
```

Programul `aestoolp.c` ilustrează modul de utilizare a bibliotecii *GNU C/C++*.

Restricție

- Numărul s al cifrelor hexazecimale semnificative din orice cheie este cuprins între 1 și 5 ($1 \leq s \leq 5$).

Precizări

- Au fost 10 de teste; pentru rezolvarea primului test s-a acordat cel mult un punct, iar pentru celelalte nouă s-au acordat cel mult câte 11 puncte.
- Un program eficient poate reconstitui cheile pentru orice fișier corect de intrare în cel mult 10 secunde.

P060106: Depozit

O firmă finlandeză de *high technology* are un depozit dreptunghiular uriaș. La depozit lucrează un muncitor și un director. Laturile depozitului sunt denumite, în ordine, stânga, sus, dreapta și jos. Suprafața depozitului este împărțită în pătrățele de dimensiuni egale, formând linii și coloane. Liniile sunt numerotate de sus în jos cu numerele întregi (1, 2, ...), iar coloanele sunt numerotate de la stânga la dreapta folosind tot numerele întregi (1, 2, ...).

Depozitul conține containere, care păstrează dispozitive cu o valoare inestimabilă. Containerele au numere de identificare distincte. Fiecare container ocupă un pătrățel. Depozitul este suficient de mare pentru ca numărul de containere care intră în depozit să fie mai mic decât numărul de linii, precum și decât numărul de coloane. Containerele nu părăsesc niciodată depozitul dar, uneori, poate sosi un nou container. Intrarea în depozit se află în colțul din stânga sus.

Muncitorul a aranjat containerele în depozit într-un anumit mod, astfel încât să le poată găsi cu ajutorul numărului de identificare. În acest scop el folosește strategia care va fi descrisă în continuare.

Să presupunem că numărul de identifcare al următorului container care sosește este k (îl vom numi containerul k). Muncitorul parcurge mai întâi prima linie, începând din partea stângă și caută primul container care are un număr de identificare mai mare decât k. Dacă un astfel de container nu există, atunci containerul k este plasat în poziția imediat următoare celui mai din dreapta container care se află deja plasat pe acea linie. Dacă găsește un container l pentru care $l > k$, atunci containerul l este înlocuit cu containerul k, iar containerul l este plasat pe linia următoare folosind aceeași metodă. Dacă muncitorul ajunge pe o linie care nu conține nici un container, atunci containerul este plasat în cel mai din stânga pătrățel al acelei linii.

Să presupunem că la intrarea în depozit au ajuns containerele 3, 4, 9, 2, 5 și 1 (în această ordine). În acest caz, rezultă următoarea repartizare a lor în depozit:

```
1 4 5
2 9
3
```

Directorul și muncitorul poartă următorul dialog:
Director: Containerul 5 a sosit înaintea containerului 4?
Muncitor: Nu, este imposibil.
Director: Deci, ești capabil să îmi spui ordinea sosirii containerelor în funcție de poziția lor în depozit.
Muncitor: În general, nu. Dar pentru acest exemplu, containerele care se află acum în depozit ar fi putut sosi în ordinea 3, 2, 1, 4, 9, 5, în ordinea 3, 2, 1, 9, 4, 5 sau în alte 14 moduri posibile.

Deoarece directorul nu vrea să pară mai puțin capabil decât muncitorul, el pleacă. Va trebui să îl ajutați pe director printr-un program care, plecând de la o poziționare a containerelor în depozit, determină toate variantele posibile în care acestea ar fi putut ajunge la depozit.

Intrare

Numele fișierului de intrare este `depot.in`. Prima linie conține un număr întreg R care reprezintă numărul de linii din depozit care conțin containere. Următoarele R linii conțin informații despre containerele de pe cele R linii începând cu latura de sus. Fiecare linie începe cu un număr întreg M care reprezintă numărul containerelor de pe linia respectivă. Urmează M numere întregi care reprezintă numerele de identificare ale containerelor de pe linia respectivă începând din partea stângă. Toate numerele de identificare I ale containerelor satisfac condiția $1 \leq I \leq 50$. Dacă N este numărul de containere din depozit, atunci $1 \leq N \leq 13$.

Ieșire

Numele fișierului de ieșire este `depot.out`. Numărul liniilor din fișierul de ieșire este egal cu numărul de posibilități de sosire a containerelor. Fiecare dintre aceste linii conțin N numere întregi care reprezintă numerele de iden-

tificare ale containerelor conform ordinii de sosire care este descrisă de acea linie. Toate liniile descriu ordini de sosire distincte.

Exemple

depot.in
```
3
3 1 4 5
2 2 9
1 3
```

depot.out
```
3 2 1 4 9 5
3 2 1 9 4 5
3 4 2 1 9 5
3 2 4 1 9 5
3 2 9 1 4 5
3 9 2 1 4 5
3 4 2 9 1 5
3 4 9 2 1 5
3 2 4 9 1 5
3 2 9 4 1 5
3 9 2 4 1 5
3 4 2 9 5 1
3 4 9 2 5 1
3 2 4 9 5 1
3 2 9 4 5 1
3 9 2 4 5 1
```

depot.in
```
2
2 1 2
1 3
```

depot.out
```
3 1 2
1 3 2
```

Punctaj

- Dacă fișierul de ieșire nu conține nimic sau conține ordini de sosire imposibile, atunci nu se va acorda nici un punct pentru testul respectiv.
- În caz contrar punctajul este calculat așa cum se descrie în continuare.
  - ♦ Dacă fișierul de ieșire conține toate ordinile de sosire posibile o singură dată, atunci veți obține patru puncte.
  - ♦ Dacă fișierul de ieșire conține cel puțin jumătate dintre ordinile de sosire posibile o singură dată, atunci veți obține două puncte.
  - ♦ Dacă fișierul de ieșire conține mai puțin de jumătate din ordinile de sosire posibile sau unele se repetă, atunci veți obține un punct.

Precizări

- Timpul maxim de execuție admis este de 0.3 secunde/test.
- Limita maximă a memoriei care poate fi folosită este de 32 MB.
- Au fost 25 de teste, pentru rezolvarea fiecăruia acordându-se cel mult 4 puncte.
- Opțiunile de compilare folosite pentru limbajul *Pascal* au fost `-So -O2 -XS`; opțiunile de compilare folosite pentru limbajele *C* și *C++* au fost `-O2 -static`.

România la CEOI și BOI

Chiar dacă acest număr al revistei este dedicat Olimpiadei Internaționale de Informatică, nu putem să nu amintim și rezultatele obținute de elevii români la celelalte concursuri internaționale care s-au desfășurat în această vară.

Olimpiada de Informatică a Europei Centrale (CEOI) a avut loc în perioada 10-17 august la Zalaegerszeg, Ungaria. Au participat concurenți din 14 țări. Delegația României a fost condusă de lect. univ. Stelian Ciurea (Universitatea "Lucian Blaga" Sibiu) și prof. Doru Popescu Anastasiu (Colegiul Național "Radu Greceanu", Slatina).

La acest concurs au participat elevii clasați pe locurile 5-8 în urma barajelor de selecție. Concurenții români au avut rezultate meritorii:

- Daniel Dumitran (București): medalie de argint (prima după cele de aur...);
- Florin Ghețu (Focșani): medalie de argint;
- Victor Costan (București): medalie de bronz;
- Vencel Bors (Oradea): foarte aproape de medalii.

În paralel s-a desfășurat și "CEOI by Net". Dintre concurenții români menționăm pe Mugurel Andreica (București) și Liviu Lalescu (Craiova), care au obținut medalii de argint.

* * *

Olimpiada Balcanică de Informatică s-a desfășurat între 8 și 13 septembrie în Durres, Albania. Au participat concurenți din 7 țări. Delegația României a fost condusă de lect. univ. Ovidiu Domșa (Universitatea "1 Decembrie 1918" Alba Iulia) și prof. Marinel Șerban (Liceul "Grigore Moisil", Iași).

La acest concurs au participat cei mai buni patru elevi români care nu au terminat încă liceul. Toți au obținut medalii de aur.

- Daniel Dumitran (București) - 580 puncte
- Vencel Bors (Oradea) - 560 puncte;
- Victor Costan (București) - 560 puncte
- Vlad Dascălu (Bacău) - 520 puncte

Concurentul clasat pe a cincia poziție a obținut doar 400 puncte din 600 posibile...