22-30 noiembrie 2003, București

Concursul Național de Programare "STELELE INFORMATICII"

Mihai Stroe

În perioada 22-30 noiembrie 2003 a avut loc la București prima ediție a Concursului Național de Programare "Stelele informaticii", organizat de Liceul Internațional de Informatică din București. În continuare vă prezentăm detalii referitoare la acest concurs și enunțurile celor 12 probleme propuse spre rezolvare.

La acest concurs, redacția *GInfo* a avut un reprezentant care a fost direct implicat în organizarea concursului și care a fost în măsură să descrie în detaliu întreaga acțiune.

Competiția a reunit la start unii dintre cei mai valoroși elevi care au obținut premii la diferite olimpiade.

A existat o perioadă de înscriere a participanților; au avut prioritate elevii care au obținut rezultate foarte bune la *ONI* 2003 și membrii lotului național lărgit din acest an.

Acest concurs a dat startul pregătirii pentru olimpiadele de informatică din 2004, fiind urmat de concursurile și activitatea de pregătire organizată demarate în luna decembrie: *Bursele AGORA* și *.Campion* (*punct Campion*). Împreună cu celelalte concursuri regionale și naționale deja existente se obține astfel un calendar competițional complet, care asigură condițiile obținerii unor rezultate valoroase la concursurile internaționale.

Participanții la acest concurs au fost împărțiți în două grupe de vârstă: elevi ai claselor a IX-a și a X-a, respectiv ai claselor a XI-a și a XII-a. Concursul de la clasele a IX-a și a X-a a avut loc în zilele de 22 și 23 noiembrie, iar cel de la clasele mari în 29 și 30 noiembrie. Pentru fiecare grupă au existat 2 zile de concurs. În fiecare zi au fost propuse spre rezolvare câte trei probleme, similare problemelor folosite la olimpiadele naționale de informatică; timpul de lucru a fost de 4 ore la clasele a IX-a și a X-a, respectiv 5 ore la clasele a XI-a și a XII-a.

Pentru fiecare grupă s-a folosit un set unic de subiecte, clasamentul făcându-se separat pe clase.

Primii clasați au primit premii constând în aplicații *software* oferite de *SOFTWIN* și premii în bani din partea *Liceului Internațional de Informatică* din București. S-au acordat un premiu I, un premiu II, un premiu III și între una și patru mențiuni la fiecare clasă, în funcție de punctajele obținute.

De asemenea, liceul a asigurat, din fonduri proprii, cazarea pentru o parte din concurenții din provincie, masa pentru toți concurenții și a recompensat financiar membrii juriului.

Din partea liceului, principalul organizator a fost domnul profesor *Ali Yuksel*, care a supervizat și a asigurat buna desfășurare a concursului, punând la dispoziție toate resursele necesare. Personal apreciez că organizarea a fost excelentă, lucru confirmat și de către concurenți.

Concursul s-a bucurat de sprijinul *Inspectoratului Școlar al Municipiului București.*

Juriul, condus de domnul *prof. univ. dr. Adrian Atanasiu*, a avut în componență studenți, foști olimpici: *Mihai Stroe*, *Marius Andrei*, *Mugurel-Ionuț Andreica*, *Radu Berinde* și *Bogdan Stroe*. De asemenea, din juriu a făcut parte domnul profesor *Osman Ay* de la *Liceul Internațional de Informatică din Constanța*, propunător de probleme la ultimele ediții ale etapei internaționale a concursului studențesc *ACM*.

Concurenții au avut la dispoziție sisteme de calcul performante, cu aceeași configurație și au putut rezolva proble-

mele folosind mediile de programare *Borland Pascal*, *Borland C*, *gcc* și *freepascal*. Corectarea s-a realizat pe sistemul de operare *Linux* cu ajutorul compilatoarelor *gcc* și *freepascal*; s-a folosit sistemul de evaluare de la *BOI* 2003. Pregătirea sistemelor și asistența tehnică au fost asigurate de personalul liceului, ajutat de elevi olimpici (*Alexandru Moșoi*, *Alexandru Radu*).

Concursul a fost considerat un succes. *Liceul Internațional de Informatică* își propune organizarea anuală a acestei competiții, aproximativ în aceeași perioadă. Detaliile organizării ediției următoare vor fi anunțate în timp util, înainte de organizarea *ONI* 2004.

În prima grupă de vârstă (clasele a IX-a și a X-a) au participat 22 de concurenți dintre care unul este elev în clasa a VIII-a (*Bogdan Stoica* - a obținut mențiune), 6 sunt elevi în clasa IX-a, iar 15 sunt elevi în clasa a X-a.

Clasamentul la clasa a IX-a este următorul:

Premiul 1: Valentin Stanciu (București)	***340***
Premiul 2: Cristina Stancu-Mara (București)	***200***
Premiul 3: Bogdan Ionescu (București)	***110***
Mențiune: Bogdan Stoica (București, cls. VIII)	***90***
Mențiune: Andrei Gönczi (Arad)	***80***

Clasamentul la clasa a X-a este următorul:

Premiul 1: Mircea Pașoi (Ploiești)	***460***
Premiul 2: Daniel Păsăilă (Suceava)	***430***
Premiul 3: Mihnea Giurgea (Ploiești)	***410***
Mențiune: Adrian Vladu (București)	***330***
Mențiune: Bogdan Tătuu (București)	***270***
Mențiune: Alex Florescu (București)	***260***

Pentru cea de-a doua grupă de vârstă (clasele a XI-a și a XII-a) au participat 28 de concurenți dintre care 14 sunt elevi în clasa XI-a, iar 14 sunt elevi în clasa a XII-a.

Clasamentul la clasa a XI-a este următorul:

Premiul 1: Leonard Crestez (Brăila)	***300***
Premiul 2: Iolanda Popa (Iași)	***280***
Premiul 3: Dan Fechete (Suceava)	***240***
Mențiune: Sorin Stancu-Mara (București)	***220***

Clasamentul la clasa a XII-a este următorul:

Premiul 1: Silviu Gănceanu (București)	***360***
Premiul 2: Alexandru Moșoi (București)	***310***
Premiul 3: Dan Ghinea (București)	***270***
Mențiune: Cristian Cutocheraș (București)	***250***
Mențiune: Mircea Digulescu (București)	***230***
Mențiune: Emilian Miron (Bacău)	***230***
Mențiune: Marius Nicolae (Slatina)	***220***

Menționăm că primii clasați la clasele a IX-a și a X-a au primit invitația de a participa și la concursul claselor a XI-a și a XII-a, fără a fi luați în calcul la acordarea premiilor. O parte dintre ei au obținut rezultate notabile. Astfel, *Mircea Pașoi* a obținut 270 puncte, *Daniel Păsăilă* - 250 puncte, iar *Bogdan Tătuu* - 220 puncte (pentru a face comparație cu premianții claselor a XI-a și a XII-a).

Prezentăm în continuare problemele propuse spre rezolvare la acest concurs, urmând ca în numărul următor să vă oferim indicații de rezolvare pentru aceste probleme.

Clasele a IX-a și a X-a

P080301: Cuvinte

Mihai Stroe

Doi prieteni, *Marius* și *Andrei*, s-au gândit la un joc.

Marius scrie pe o foaie un șir de *N* numere. Sub fiecare număr, el scrie câte o literă: sub primul număr litera *A*, sub al doilea număr litera *B*, și tot așa, în ordine lexicografică. *Marius* și *Andrei* folosesc un alfabet cu câteva mii de litere, cunoscut numai de ei și care începe cu literele de la *A* la *Z* (pentru a putea fi folosit și în comunicarea cu alți oameni). Literele sunt deci folosite ca indici pentru numerele din șir.

Andrei caută apoi toate "cuvintele" posibile care respectă următoarele condiții:

- un cuvânt reprezintă un șir de litere, ordonat lexicografic;
- numerele din șir, corespunzătoare literelor dintr-un cuvânt și scrise în ordinea dată de acestea, sunt în ordine strict crescătoare.

De exemplu, pentru șirul 2 1 3 5 4, scriind dedesubt literele *A B C D E*, câteva dintre cuvintele valide sunt *AC*, *ACD*, *ACE*, *AD*, *BE* etc., dar *AB*, *ED* sau *BDE* nu sunt cuvinte valide.

Apoi, *Andrei* alege dintre aceste cuvinte pe cele de lungime maximă și le scrie în ordine lexicografică. Pentru exemplul de mai sus, acestea sunt *ACD*, *ACE*, *BCD* și *BCE*. Dintre aceste cuvinte de lungime maximă, el i-l spune lui *Marius* pe al *K*-lea în ordine lexicografică. Dacă *Andrei* spune corect (și repede) cuvântul acesta, el câștigă jocul și îl pierde în caz contrar.

Andrei câștiga întotdeauna și era foarte mândru de el, până într-o bună zi, când *Marius* i-a scris un șir de 67 de numere... I-ar fi fost cam greu lui *Andrei* să facă tot ce făcea de obicei, așa că s-a gândit să scrie un program care să-i dea direct rezultatul. Pentru asta, a apelat la un profesionist.

Scrieți un program care determină cuvântul cerut și îl ajută pe *Andrei* să câștige jocul.

Date de intrare

Fișierul de intrare `cuvinte.in` conține pe prima linie două numere *N* și *K*, separate printr-un spațiu, reprezentând numărul de numere din șir, respectiv numărul de ordine al cuvântului cerut. Pe a doua linie se află *N* numere întregi separate prin câte un spațiu, numerele scrise de *Marius* pe foaie.

Date de ieșire

În fișierul `cuvinte.out` se va scrie cuvântul cerut. Deoarece nu se cunosc literele care urmează după *Z* în alfabetul celor doi, în locul literelor cuvântului se vor scrie numerele

de ordine ale acestora în alfabet, separate prin câte un spațiu. Astfel, de exemplu, cuvântul *ACZ* ar fi scris ca 1 3 26.

Deci, în fișier se vor scrie, separate printr-un spațiu, numere întregi care reprezintă numerele de ordine în alfabet ale literelor celui mai lung cuvânt cu numărul de ordine *K*, format după regulile din enunț.

Restricții și precizări

- N este număr natural și $1 < N \leq 200$;
- K este număr natural și $1 \leq K \leq 2.000.000.000$;
- numerele din șir sunt întregi cuprinse între 0 și 10000 inclusiv;
- pentru orice set de date de test care va fi folosit vor exista cel puțin K și cel mult 2.000.000.000 de cuvinte de lungime maximă care se pot forma respectând regulile descrise.

Exemplu

cuvinte.in

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

cuvinte.out

```
2 3 4
```

Explicație

Numerele din fișierul de ieșire corespund pozițiilor 2, 3 și 4 din șir pe care se află numerele 1, 3, respectiv 5.

Timp de execuție: 0,1 secunde/test

P080302 Timp

Marius Andrei

Avem la dispoziție o clepsidră mai ciudată cu care trebuie să măsurăm o anumită perioadă de timp.

Clepsidra este mai ciudată prin faptul că nu are doar două compartimente unde se poate strânge nisipul, ci trei.

Clepsidra se sprijină pe două dintre aceste compartimente, iar din al treilea se scurge nisipul în mod egal în celelalte. La început avem tot nisipul adunat într-un singur compartiment, iar cu el putem măsura, să presupunem 16 minute, ca în figura 1 a).

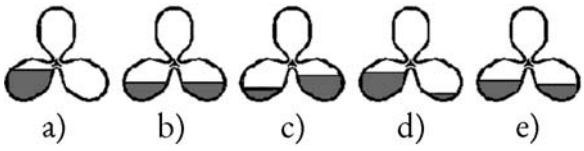

Figura 1

Pentru a măsura 7 minute în unul dintre compartimente putem roti clepsidra cu 120° în sensul acelor de ceasornic sau invers. Considerăm că rotația spre dreapta se efectuează în sensul acelor de ceasornic, iar rotația spre stânga se efectuează invers acelor de ceasornic. De exemplu, rotind spre dreapta obținem figura 1 b), în care ambele compartimente conțin nisip pentru a măsura 8 minute. Rotim spre stânga și obținem 4 și 12. Rotim spre dreapta și obținem 14 și 2, după care rotim spre dreapta și obținem 9 și 7. Astfel am obținut 7 într-un compartiment și putem măsura acest timp.

Scrieți un program care determină o serie de rotiri pentru a putea măsura un interval timp.

Date de intrare

Fișierul de intrare **timp.in** conține două numere întregi N și K separate între ele printr-un singur spațiu. N reprezintă timpul care poate fi măsurat inițial, tot nisipul aflându-se în stânga-jos (ca în exemplu), iar K reprezintă timpul pe care vrem să-l obținem într-un compartiment. Nu contează în ce parte se află nisipul cu care măsurăm timpul final.

Date de ieșire

În fișierul **timp.out** se va scrie pe prima linie numărul de rotiri necesare pentru a obține cantitatea cerută de nisip. Pe următoarele linii se va scrie câte o valoare care reprezintă rotirile în ordinea în care au fost efectuate. Pentru o rotație spre stânga se va scrie 0, iar pentru o rotație spre dreapta se va scrie 1.

Restricții și precizări

- $1 < K < N < 200.000.001$;
- soluția nu este unică și numărul de transformări nu trebuie să fie minim;
- după o rotație tot nisipul din partea de sus trebuie să se scurgă.

Exemplu

timp.in

```
16 7
```

timp.out

```
4
1
0
1
1
```

Timp de execuție: 0,1 secunde/test

P080303: Vecini

Marius Andrei

În *Drumul Taberei* există un bloc foarte ciudat. În primul an când a fost construit (să presupunem anul 1) avea un singur etaj, după care în fiecare an se construiește un nou etaj, astfel încât în anul X va avea X etaje.

Dar acesta nu este singurul lucru ciudat. Și modul în care este ocupat fiecare etaj este foarte ciudat.

La etajul 1 (primul etaj începând numărătoarea de jos) stă tot timpul administratorul, deci este ocupat. De asemenea ultimul etaj, fiind nou, este tot timpul ocupat. Restul etajelor însă sunt ocupate (sau libere) după regulile:

- dacă anul trecut etajul curent și etajul de dedesubt au fost ocupate, atunci etajul va fi liber anul acesta;

- dacă anul trecut etajul curent și etajul de dedesubt au fost libere, atunci etajul va fi liber și anul acesta;
- dacă anul trecut etajul curent a fost ocupat, iar cel de dedesubt a fost liber, atunci etajul curent va fi ocupat;
- dacă anul trecut etajul curent a fost liber, iar cel de dedesubt a fost ocupat, atunci etajul curent va fi ocupat.

Scrieți un program care determină configurația etajelor după *N* ani de la construcție.

Date de intrare
Fișierul de intrare **vecini.in** conține un singur număr natural *N*, care reprezintă numărul de ani.

Date de ieșire
În fișierul **vecini.out** se vor scrie *N* numere separate între ele printr-un spațiu. Numerele reprezintă starea fiecărui etaj, începând cu etajul 1. Reprezentarea este 0 pentru liber și 1 pentru ocupat.

Restricție
- $1 < N < 100.001$.

Exemplu
vecini.in
```
5
```

vecini.out
```
1 0 0 0 1
```

Timp de execuție: 0,2 secunde/test

P080304: Editor

Mugurel-Ionuț Andreica

Se consideră un editor de text care răspunde doar la apăsarea a șase taste, și anume cele care au tipărite simbolurile: "(", ")", "[", "]", "*" și "E". Dacă se apasă una din tastele "(", ")", "[", "]", atunci se afișează caracterul respectiv pe ecran. Dacă se apasă tasta "*" se șterge ultimul caracter afișat (dacă nu este afișat nici un caracter, atunci nu se întâmplă nimic). Dacă se apasă tasta "E" (*Enter*), atunci editorul va verifica dacă șirul afișat pe ecran este un șir parantezat corect. Un șir este parantezat corect dacă este construit conform regulilor:
- <*șir parantezat corect*> = <*șirul vid*>;
- <*șir parantezat corect*> = "(" + <*șir parantezat corect*> + ")";
- <*șir parantezat corect*> = "[" + <*șir parantezat corect*> + "]";
- <*șir parantezat corect*> = <*șir parantezat corect* 1> + <*șir parantezat corect* 2>.

Prin *X* + *Y* s-a notat concatenarea șirurilor *X* și *Y*.

De exemplu, șirurile "[]([([])])", "[[[([[[]]])]]](())" și "([([])[()]])[]" sunt parantezate corect, iar șirurile ")[](", "[()()()()[[[(" și "[(()[([]())]])" nu sunt parantezate corect.

Dându-se o succesiune de taste apăsate care se termină cu tasta "E" și știind că, inițial, nu este nici un caracter afișat pe ecran, trebuie să decideți dacă șirul afișat pe ecran în urma apăsării tastelor este un șir parantezat corect.

Date de intrare
Pe prima linie a fișierului de intrare **editor.in** se află un număr întreg *T*, reprezentând numărul de succesiuni de taste care vor fi descrise în continuare. Pe fiecare din următoarele *T* linii este descrisă câte o succesiune de taste apăsate.

Date de ieșire
În fișierul **editor.out** veți scrie exact *T* linii, câte una pentru fiecare succesiune de taste descrisă în fișierul de intrare.

Pe fiecare linie veți afișa ":)" (fără ghilimele) dacă șirul obținut în urma succesiunii corespunzătoare de taste apăsate este parantezat corect , respectiv ":(" (fără ghilimele) în caz contrar.

Restricții și precizări
- $1 \le T \le 30$;
- nici o succesiune de caractere nu va conține mai mult de 60000 de taste apăsate;
- nu se acordă punctaje parțiale.

Exemplu
editor.in
```
6
E
)))))*****]]]]]*****E
(([[[()[()[([])][[]()([])](())]])([
][][])E
**[*]*[]()[]*(**)*E
([](([]))E
[E
```

editor.out
```
:)
:)
:)
:)
:(
:(
```

Timp de execuție: 1 secundă/test

P080305: Numere

Bogdan Stroe

Ionel are o slăbiciune pentru jocurile de perspicacitate. El încearcă să rezolve un astfel de joc în care are o matrice de dimensiune $M \times N$ ale cărei elemente sunt numere întregi.

O mutare constă în scăderea dintr-un element a numărului de vecini ai poziției elementului respectiv și creșterea cu unu a valorilor din pozițiile vecine.

Vecinii pentru o anumită poziție sunt elementele de pe orizontală și verticală, deci o poziție poate avea 2, 3 sau 4

vecini, după cum este situată într-un colț, pe o latură sau în interiorul matricei.

De exemplu, având matricea cu două linii și trei coloane:

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

o mutare la linia 1 și coloana 2 ar duce la matricea:

```
2 -1 5
4  3 5
```

Anumite poziții cunoscute din matrice sunt *sensibile la scădere*, adică nu se pot efectua mutări în pozițiile respective. Numărul acestor poziții este egal cu *K*.

Scopul lui *Ionel* este ca, pornind de la matricea inițială, să obțină o matrice în care toate elementele să aibă aceeași paritate, efectuând un număr minim de mutări.

Date de intrare

Pe prima linie a fișierului de intrare **numere.in** se află trei numere întregi *M*, *N* și *K*, cu semnificațiile din enunț.

Pe următoarele *M* linii se află câte *N* numere întregi cuprinse între 1 și 10000 inclusiv, reprezentând matricea inițială.

Pe următoarele *K* linii se află câte două numere întregi separate prin spațiu, reprezentând coordonatele pozițiilor *sensibile la scădere* (prima coordonată corespunde liniei, iar cea de-a doua coloanei).

Date de ieșire

În fișierul de ieșire **numere.out** se va afișa pe prima linie numărul minim de mutări *MIN* care rezolvă jocul.

Pe următoarele *MIN* linii se vor afișa mutările, câte o mutare pe linie (două numere întregi separate printr-un spațiu, reprezentând linia și coloana poziției fiecărei mutări).

În cazul în care nu există soluție, în fișierul de ieșire se va scrie, pe un singur rând, mesajul: `NU EXISTA SOLUTIE`.

Restricții și precizări

- $2 \le M, N \le 12$;
- $0 \le K \le M \cdot N$;
- elementele din matrice sunt numere întregi cuprinse între 1 și 10000 inclusiv; pe parcursul jocului, este permis ca elementele din matrice să aibă valori în afara acestui interval;
- dacă există mai multe soluții cu număr minim de mutări, atunci se va afișa una dintre ele.

Exemplu

numere.in

```
3 3 1
0 3 0
2 4 -4
9 8 3
3 1
```

numere.out

```
2
1 2
2 2
```

Timp de execuție: 1 secundă/test

P080306: Oo

Osman Ay

Fermierul *Ion* are o fermă de formă circulară, unde cresc *N* găini. Ferma a fost împărțită în *N* sectoare, numerotate de la 1 la *N*, astfel încât oricare două sectoare având numere consecutive sunt adiacente (se află unul lângă altul). În plus, primul și ultimul sector sunt adiacente. În fiecare sector se află câte o găină, iar aceasta depune un anumit număr de ouă în fiecare zi.

După ce găinile depun ouăle, fermierul *Ion* dorește să le adune, pentru a le mânca. Deoarece fermierul este foarte lacom, de fiecare dată el alege două sectoare adiacente din care adună ouale simultan.

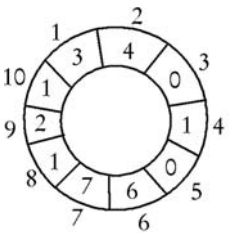

Figura 2: Ferma

Din păcate, din cauza lăcomiei sale, găinile din sectoarele vecine cu cele două alese se sperie și devin violente, motiv pentru care fermierul nu mai poate aduna ouăle din aceste sectoare. În exemplul din figură, dacă fermierul adună ouăle din sectoarele 1 și 2, el nu va mai putea aduna ouăle din sectoarele 3 și 10.

Determinați numărul maxim de ouă pe care le poate aduna fermierul *Ion*, în urma aplicării strategiei sale lacome.

Date de intrare

Fișierul de intrare **oo.in** conține pe prima linie numărul de sectoare în care este împărțită ferma (și, implicit, numărul de găini). Pe următoarea linie se află *N* numere întregi, reprezentând numărul de ouă depuse de fiecare găină, în ordinea sectoarelor în care se află acestea.

Date de ieșire

În fișierul de ieșire **oo.out** se va afișa un număr întreg care reprezintă numărul maxim de ouă pe care le poate aduna fermierul *Ion*.

Restricții și precizări

- $2 \le N \le 10.000$;
- numărul de ouă pe care le poate depune o găină în fiecare zi este un număr întreg cuprins între 0 și 100 inclusiv.

Exemplu

oo.in

```
10
3 4 0 1 0 6 7 1 2 1
```

oo.out

```
20
```

Explicație
Fermierul poate aduna ouăle din sectoarele 2 și 3 (4+0), 6 și 7 (6+7) și 9 și 10 (2+1).

Timp de execuție: 0,1 secunde/test

Clasele a XI-a și a XII-a

P080307: Arbori

Mihai Stroe

Am doi copaci reprezentați prin doi arbori binari cu M, respectiv N noduri. În fiecare nod al unui astfel de arbore cresc flori. Ordinea ramurilor (fiilor) contează (se face distincție între fiul stâng și fiul drept).

Doi arbori binari cu flori în noduri sunt similari dacă (definiție recursivă):
- au ambii 0 noduri, sau
- au ambii cel puțin un nod, același număr de flori în rădăcină, subarborii stângi sunt similari și subarborii drepți sunt similari.

Vreau să transform cei doi arbori dați în doi arbori similari (conform definiției de mai sus) cu exact K flori fiecare, având aceleași rădăcini cu arborii inițiali. Pentru a-mi atinge scopul pot să fac două tipuri de operații:
- tai o ramură (elimin un subarbore din unul dintre cei doi arbori);
- rup o floare (scad cu 1 numărul de flori din unul din nodurile unuia dintre arbori).

Deoarece vreau să muncesc cât mai putin pentru a-mi atinge scopul, doresc să tai cât mai puține ramuri și, dacă am mai multe variante de a tăia cât mai puține ramuri, să aleg una în care să rup cât mai puține flori.

Scrieți un program care să mă ajute.

Date de intrare
Fișierul de intrare **arbori.in** conține pe prima linie numărul K de flori care trebuie să rămână în cei doi arbori. Urmează cei doi arbori, unul după altul.

Pentru fiecare arbore, pe prima linie se află numărul de noduri NR. Urmează NR linii, fiecare conținând informația pentru un nod: numărul de flori F, numărul fiului stâng (sau 0 dacă acesta nu există) și numărul fiului drept (sau 0 dacă acesta nu există).

Arborii sunt valizi (conțin numerele de la 1 la NR, nu au cicluri etc.) și au amândoi rădăcina în nodul 1.

Date de ieșire
În fișierul **arbori.out** se va afișa pe prima linie numărul T de tăieri de ramuri și numărul R de ruperi de flori din soluția optimă. Pe următoarele T linii veți afișa tăierile de ramuri. Tăierea unei ramuri se reprezintă prin două numere ARB și NOD; aceasta înseamnă că am tăiat ramura din arborele ARB (1 sau 2), care lega nodul NOD de tatăl lui. Pe următoarele R linii veți afișa ruperile de flori. Ruperea unei flori se reprezintă prin două numere ARB și NOD; aceasta înseamnă că am rupt o floare din nodul NOD al arborelui ARB. Numerele de pe fiecare linie se vor separa printr-un spațiu. Dacă există mai multe soluții optime, afișați una oarecare. Toate testele date vor avea cel puțin o solutie.

Restricții
- $1 \le M, N \le 100$;
- $0 \le F \le 10$;
- $0 \le K \le 100$;
- $1 \le NR \le 100$.

Exemplu

arbori.in
```
7
3
5 2 3
2 0 0
1 0 0
3
6 2 0
6 0 3
5 0 0
```

arbori.out
```
2 5
1 3
2 3
2 1
2 2
2 2
2 2
2 2
```

Timp de execuție: 1 secundă/test

P080308: Sum

Marius Andrei

Se dă un șir de numere întregi. Se caută un subșir cu lungimea cuprinsă între L și U, format din elemente consecutive ale șirului inițial, cu suma elementelor maximă.

Scrieți un program care determină suma maximă a unui astfel de subșir.

Date de intrare
Fișierul de intrare **sum.in** conține pe prima linie trei numere întregi N, L și U, despărțite între ele printr-un spațiu. N reprezintă lungimea șirului mare. L este lungimea minimă și U este lungimea maximă. Pe următoarele linii se află cele N numere, despărțite prin câte un spațiu.

Date de ieșire
În fișierul **sum.out** se va scrie pe prima linie suma maximă care se poate obține.

Restricții și precizări
- $1 \le L \le U \le N \le 100.001$;
- numerele din șir sunt numere întregi și au valori din intervalul [-10 000, 10 000].

Exemplu
sum.in
```
9 2 3
100 -100 0 10 -5 0 10 0 1
```

sum.out
```
11
```

Timp de execuție: 0,1 secunde/test

P080309: Taxi

Marius Andrei

Orașul *X*3 a fost gândit de informaticieni, astfel încât străzile să împartă orașul sub forma unei matrice. Intersecțiile dintre străzi sunt practic punctele cu coordonate întregi cuprinse în intervalul [0, *A*] pentru coordonata *X* și [0, *B*] pentru coordonata *Y*.

În acest oraș există exact două taxiuri. Pentru că s-au înțeles între ei, ambii taximetriști percep același tarif pentru aceeași distanță parcursă. Așa că și oamenilor le este indiferent cu care dintre taxiuri se deplasează. Când este nevoie de taxi, singurul criteriu este taxiul care este mai aproape. Cetățenii sunt foarte bine informați și cunosc la ce coordonate se află taxiurile. Nu le rămâne decât să decidă care este mai aproape. Probleme serioase apar atunci când taxiurile sunt egal depărtate pentru că cetățenii nu se pot hotărî și rămân pe loc ore întregi.

Azi fiind o zi foarte aglomerată, în fiecare intersecție se află exact un om care dorește să apeleze la un taxi.

Scrieți un program care determină câți oameni vor rămâne nehotărâți.

Date de intrare
Fișierul de intrare **taxi.in** conține pe prima linie numărul de teste *T*.

Pe următoarele *T* linii se află câte 6 numere întregi, separate printr-un spațiu reprezentând *A* (coordonata maximă pentru *X*), *B* (coordonata maximă pentru *Y*) și coordonatele celor două taxiuri, și anume *x*1 *y*1 *x*2 *y*2.

Date de ieșire
În fișierul **taxi.out** se vor scrie *T* linii, câte una pentru fiecare test. Pe fiecare linie se va scrie un număr întreg, reprezentând numărul de persoane indecise pentru testul respectiv.

Restricții și precizări
- $0 < T < 101$;
- $0 < A, B < 10.001$;
- $0 \le x_1, x_2 \le A$ și $0 \le y_1, y_2 \le B$.

Exemplu
taxi.in
```
1
9 9999 0 0 2 0
```

taxi.out
```
10000
```

Timp de execuție: 0,1 secunde/test

P080310: Expr

Radu Berinde

Acarie, un student eminent, are probleme cu tema pentru seminarul de structuri algebrice. A redus problema la una mai simplă, care constă în operații cu mulțimi. Însă sunt prea multe operații și deja este timpul să plece în oraș. Ajutați-l să iasă din încurcătură.

Scrieți un program care găsește rezultatul unei expresii valide date.

Date de intrare
În fișierul de intrare **expr.in** se află un șir de caractere fără spații. Acest șir de caractere poate conține următoarele elemente:
- operanzii - sunt mulțimi, descrise de o acoladă "`{`" urmată de numerele care fac parte din mulțimea respectivă (în ordine crescătoare) despărțite de "`,`" (virgulă) și terminate cu o acoladă "`}`";
- operatorii - care pot fi paranteze, sau una din următoarele operații (care au aceeași prioritate):
  - `*` - intersecție;
  - `+` - reuniune;
  - `-` - diferență
  - `%` - diferență simetrică: `A % B = (A - B) + (B - A) = (A + B) - (A * B)`.

Date de ieșire
Scrieți rezultatul expresiei în fișierul **expr.out** ca o singură mulțime (după formatul mulțimilor din fișierul de intrare).

Restricții și precizări
- lungimea șirului din fișierul de intrare este mai mică sau egală cu 100.000;
- numărul valorilor distincte care apar în expresie este mai mic sau egal cu 8.000;
- valorile care apar în mulțimi sunt numere întregi din intervalul [0, 2.000.000.000];
- adâncimea maximă de parantezare nu depășește valoarea 100;
- numărul maxim de operații care apar în expresie este mai mic sau egal cu 10.000;
- mulțimea vidă este reprezentată prin "`{}`";
- numerele din mulțimea rezultată vor fi afișate în ordine crescătoare.

Exemplu

expr.in

```
{1,2,3,4}%({1,2,3,4}*{}+{1,2}+{5,6}-{1})
```

expr.out

```
{1,3,4,5,6}
```

Timp de execuție: 0,2 secunde/test

P080311: Ciclu

Radu Berinde

Acarie, fericit că a trecut examenul la algebră pleacă în oraș să sărbătorească. Este atât de fericit, încât nu se mulțumește cu un simplu traseu între terase, ci vrea un ciclu pentru a fi sigur că poate continua să petreacă atâta timp cât dorește. Din păcate, drumurile dintre terase au costuri diferite și *NU* sunt bidirecționale. *Acarie* nu este interesat neapărat să minimizeze costul acestui ciclu, ci dorește să minimizeze costul *MEDIU* al ciclului (evident, vrea să meargă cât mai puțin între terase, astfel încât costul ciclului împărțit la numărul de drumuri dintre terase să fie minim).

Ajutați-l și de această dată pe *Acarie* și vă va cinsti toată ziua. Scrieți un program care determină costul mediu minim al unui ciclu de terase.

Date de intrare

În fișierul de intrare **ciclu.in** se află N (numărul de terase), M (numărul de drumuri unidirecționale) și cele M drumuri directe dintre terase. Acestea sunt scrise fiecare pe câte o linie, reprezentate de trei numere întregi - terasa de unde pornește drumul, terasa unde ajunge drumul (numere între 1 și N) și costul drumului (întreg strict pozitiv mai mic decât 100000).

Date de ieșire

Să se afișeze în fișierul **ciclu.out** costul mediu minim al ciclului găsit, cu două zecimale exacte.

Restricții

- $2 \le N \le 1000$;
- $2 \le M \le 4000$;
- există cel puțin un ciclu.

Exemplu

ciclu.in

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

ciclu.out

```
1.75
```

Timp de execuție: 3 secunde/test

P080312: Prefix

Mugurel-Ionuț Andreica

Se consideră un șir format din literele mici 'a', 'b', …,'z'. Să se determine cel mai lung prefix periodic al său. Un șir X este periodic dacă se poate scrie sub forma $P + P + \ldots + P$, unde prin $A + B$ s-a notat concatenarea șirurilor A și B. Șirul P se numește perioada lui X și trebuie să fie strict mai scurt decât X.

De exemplu, șirurile "abcaabcaabcaabca", "xyyxyy" și "wwwww" sunt periodice, iar șirurile "abcaabcaabcaz", "xyxyxz" și "wwwaawww" nu sunt periodice.

Date de intrare

Prima linie a fișierului de intrare **prefix.in** conține numărul întreg T de șiruri prezente în fișier. Fiecare din următoarele T linii conține câte un șir.

Date de ieșire

În fișierul de ieșire **prefix.out** se va afișa, pentru fiecare din cele T șiruri din fișierul de intrare, lungimea celui mai lung prefix periodic.

Restricții și precizări

- $1 \le T \le 10$;
- fiecare șir va avea cel puțin unul și cel mult 1.000.000 de caractere;
- nu se acordă punctaje parțiale.

Exemplu

prefix.in

```
10
abcdefgh
z
xxxuxxxu
abbcaabbcaabbcaabbcaxyzxyzxyzxyz
hellohellohellohellauhellohello
aaaaaaaaazaaaaaaaaazaaaaaaaaa
uvwuvwuvwuvwu
sirperiodicsirperiodicsirperiodicsir
aababcabcdabcdeabcdefabcdefgaerror
mamatatabunicubunicaunchiumatusa
```

prefix.out

```
0
0
8
20
15
20
12
33
2
4
```

Timp de execuție: 1,5 secunde/test