ACM

În perioada 16-19 octombrie a avut loc la București ediția 2003/2004 a concursului internațional de programare ACM, la care au participat echipe reprezentând țări din sud-estul Europei. În paginile următoare vă prezentăm enunțurile problemelor propuse spre rezolvare la acest concurs.

Facultatea de Automatizări și Calculatoare din cadrul Universității "Politehnica" București a găzduit și ediția din acest a faie locale a concursului *ACM*, la care au participat echipe reprezentând țări din *sud-estul Europei*.

Pentru acest concurs s-au înscris 60 de echipe din *Bulgaria*, *Bosnia* și *Herțegovina*, *Grecia*, *Macedonia*, *Moldova*, *România*, *Ucraina* și *Iugoslavia*.

Dintre cele 60 de echipe au fost acceptate pentru concurs 53 de echipe de la 44 de universități din cele opt țări.

Echipele au avut la dispoziție cinci ore pentru a rezolva nouă probleme, folosind unul din limbajele de programare ***C/C++***, ***Pascal*** sau ***Java***. Gradul de dificultate al problemelor a fost relativ ridicat.

Calculatoarele puse la dispoziția participanților au avut procesoare *Pentium IV* cu frecvența de 2,6 *GHz*, 256 *MB* de memorie *RAM* și *harddisk-uri* cu o capacitate de 60 *GB*. Acestea au fost interconectate într-o rețea. Sistemele de operare puse la dispoziție au fost Windows *XP* și *Linux*.

Trei dintre echipele participante au reușit să rezolve 7 probleme, trei echipe au rezolvat 6 probleme și trei echipe au rezolvat 5 probleme. Majoritatea echipelor au rezolvat cel mult 3 probleme.

Primul loc a fost obținut de echipa a doua de la *Universitatea "St. Kliment Ohridski"* din *Sofia*, *Bulgaria*. Echipa l-a avut ca îndrumător pe *Krassimir Manev* și a fost formată din *Kristiyan Haralambiev*, *Miloslav Sredkov* și *Petko Minkov*. Aceștia au reușit să rezolve șapte probleme cu o penalizare de 827 de puncte.

Locul al doilea a fost obținut de echipa a treia de la aceeași universitate și care a avut același îndrumător. Echipa a fost formată din *Anev Ivan*, *Riskov Ivaylo* și *Tsvetan Bogdanov*. Aceștia au reușit să rezolve tot șapte probleme, dar cu o penalizare de 922 de puncte.

Locul al treilea a fost obținut de echipa "*UNIBUC-COLEGIU*" de la *Universitatea București*. Echipa l-a avut ca îndrumător pe *Cristian Grozea* și a fost formată din *Angel-Stelian Proorocu*, *Matei Gruber* și *Tiberiu Dănet*. Și ei au rezolvat șapte probleme, dar penalizarea a fost de 1085 de puncte.

Primele trei echipe, membrii juriului precum și cei care au avut grijă de buna desfășurare a concursului au primit premii de la *IBM*.

În continuare vă prezentăm enunțurile celor nouă probleme propuse spre rezolvare la acest concurs.

P070301: Codificare delta

Un algoritm simplu de criptare care reprezintă o îmbunătățire substanțială adusă metodei substituției este metoda *delta*. În continuare prezentăm modul de funcționare al acestuia.

Literelor alfabetului englezesc cuprinse între *A* și *Z* le sunt asociate numere de la 0 la 25. Valorile numerice nu sunt atașate literelor în ordinea în care apar în alfabet, dar fiecare literă trebuie să fie asociată cu un singur număr.

Metoda de codificare delta implicită asociază literelor numerele în aceeași ordine în care apar în alfabet (ordinea lexicografică), deci *A* este asociat cu 1, *B* cu 2 etc. Această asociere este utilizată atunci când nu există o alta mai bună.

Asocierea literelor cu numere cuprinse între 0 și 25 poartă numele de *cifru*.

Pentru a codifica un mesaj se folosesc *diferențele* dintre valorile succesive asociate literelor din alfabet.

Pentru codificarea unui mesaj cu *metoda delta* trebuie respectate următoarele reguli:

- toate simbolurile, în afară de litere, sunt transferate în mesajul criptat neschimbate;
- când o literă este precedată de un simbol care nu este literă sau nu este precedată de nici un simbol (cazul în care aceasta este prima literă din mesajul inițial) se consideră că este precedată de litera *A*;
- dacă o literă este majusculă, atunci va fi majusculă și în mesajul criptat, iar dacă nu este majusculă, nu va fi majusculă nici în mesajul criptat;
- când o literă este precedată de o altă literă, atunci valoarea diferenței este egală cu valoarea asociată primei litere care se scade din valoarea asociată celei de-a doua. Litera care are asociată valoarea diferenței modulo 26 o va înlocui pe cea de-a doua în mesajul criptat.

Sarcina voastră este de a cripta și decripta mesaje folosind această metodă.

Date de intrare

Fișierul de intrare **A.IN** conține pe fiecare linie câte o comandă urmată de parametrii corespunzători și fișierul se termină cu *EOF* (*End Of File - sfârșit de fișier*).

O comandă este constituită din unul dintre cuvintele: `ENCRYPT`, `DECRYPT` sau `CIPHER` urmat de cel puțin un spațiu. Modul în care sunt scrise aceste comenzi nu este important (de exemplu, `encrypt` și `enCrypt` desemnează aceeași comandă).

Comanda `ENCRYPT` primește ca parametru (este urmată de) textul care trebuie criptat folosind cifrul curent.

Comanda `DECRYPT` primește ca parametru (este urmată de) textul care trebuie decriptat folosind cifrul curent.

Pentru comenzile `ENCRYPT` și `DECRYPT`, dacă nu a fost furnizat un cifru, se va folosi cifrul implicit.

Comanda `CIPHER` trebuie să fie urmată de cele 26 de litere ale alfabetului. Ordinea în care se află aceste litere determină valorile care le vor fi asociate astfel că prima literă din acest șir va fi asociată cu valoare 0, a doua cu valoarea 1 etc. Literele pot fi separate între ele prin 0 sau mai multe spații.

În cazul în care cifrul furnizat nu este valid (de exemplu, nu apar toate literele din alfabet sau o literă apare de două ori) se va folosi cifrul implicit.

Date de ieșire

Rezultatele vor fi furnizate la ieșirea standard. Pentru fiecare linie din fișierul de intrare se va furniza o linie la ieșire astfel:

- în cazul în care o linie nu începe cu una din cele trei comenzi posibile urmată de cel puțin un spațiu, în fișierul de ieșire, pe linia corespunzătoare se va afișa mesajul "`Command not understood`";
- pentru comanda `ENCRYPT`, la ieșire, pe linia corespunzătoare se va fi scrie mesajul "`RESULT: `" urmat de textul mesajului criptat corespunzător celui primit ca parametru;
- pentru comanda `DECRYPT`, la ieșire, pe linia corespunzătoare se va fi scrie mesajul "`RESULT: `" urmat de textul mesajului decriptat corespunzător celui primit ca parametru;
- pentru comanda `CIPHER`, la ieșire, pe linia corespunzătoare se va fi scrie mesajul "`Good cipher. Using `" urmat de cifrul scris cu majuscule, dacă cifrul este valid și "`Bad cipher. Using default.`" în caz contrar.

Restricție

- nici o linie nu conține mai mult de 10.000 de caractere.

Exemplu

A.IN

```
ENCRYPT The quick brown fox leaped over the lazy dog.
CIPHER qwertyuiopasdfghjklzxcvbnm
ENCRYPT The quick brown fox leaped over the lazy dog.
DECRYPT Xsf juwgv fuyzb rcd oaomza nguw xsf olpd evu.
ENCRIPT This won't work!
CIPHER qAZwSXedCRfv  tgbyhnujmikolp
ENCRYPT The quick brown fox leaped over the lazy dog.
DECRYPT Vsh puwgb gibeb rgc ocilda kbiw vsh owag ehh.
CIPHER qwertyuopasdfghjklzxcvbnm
ENCRYPT The quick brown fox leaped over the lazy dog.
DECRYPT Tox qeoui bqxir fjj ltwppz ohjn tox lpzz dls.
```

Ieșirea standard

```
RESULT: Tox qeoui bqxir fjj ltwppz ohjn tox lpzz dls.
Good cipher. Using QWERTYUIOPASDFGHJKLZXCVBNM.
RESULT: Xsf juwgv fuyzb rcd oaomza nguw xsf olpd evu.
RESULT: The quick brown fox leaped over the lazy dog.
Command not understood.
Good cipher. Using QAZWSXEDCRFVTGBYHNUJMIKOLP.
RESULT: Vsh puwgb gibeb rgc ocilda kbiw vsh owag ehh.
RESULT: The quick brown fox leaped over the lazy dog.
Bad cipher. Using default.
RESULT: Tox qeoui bqxir fjj ltwppz ohjn tox lpzz dls.
RESULT: The quick brown fox leaped over the lazy dog.
```

P070302: Parola ascunsă

Câteodată programatorii au metode din cele mai diverse pentru a-și ascunde parolele. De exemplu, să vedem cum *Billy "Hacker" Geits* își ascunde propriile parole.

Billy își alege un șir de caractere de lungime *L* format din litere mici ale alfabetului latin. Pentru acest șir de caractere *Billy* face toate cele *L* - 1 deplasări circulare la stânga cu o poziție și le pune unele sub altele. Dintre aceste *L* - 1 șiruri astfel obținute, înaintea cărora se trece șirul inițial, se alege cel care este primul în ordine lexicografică, parola constituind-o un prefix al acestuia.

De exemplu, pentru șirul `alabala`, șirurile care se obțin prin deplasări circulare, incluzându-l și pe acesta, sunt:

```
alabala
labalaa
abalaal
balaala
alaalab
laalaba
aalabal
```

Șirul care este primul în ordine lexicografică se află pe poziția a șasea (numerotarea șirurilor se face începând cu poziția 0), și în concluzie, prima literă a acestui șir se află pe poziția a șasea din șirul inițial.

Scrieți un program care pentru un șir *S* dat determină poziția celei "mai mici"(primei) deplasări în ordine lexicografică. Dacă cel mai mic șir de caractere apare de mai multe ori, să se cere cea mai mică poziție pe care acesta începe.

Date de intrare

Fișierul de intrare **B.IN** conține pe prima linie numărul *T* al testelor.

Pe fiecare dintre următoarele *T* linii se află o pereche formată dintr-un număr întreg *L* și un șir de caractere *S* se-

parate printr-un singur spațiu. *L* reprezintă lungimea șirului de caractere.

Date de ieșire
Rezultatele se vor scrie la ieșirea standard. Rezultatul constă în *T* linii.

Pentru fiecare linie din fișierul de intrare se va scrie pe o linie un singur număr care reprezintă poziția determinată, conform enunțului problemei, pentru un șir *S* de lungime *L*.

Rezultatele pentru fiecare test se vor scrie în ordinea în care apar în fișierul de intrare.

Restricție
• $5 \le L \le 100.000$.

Exemplu

B.IN	*Ieșirea standard*
2	1
6 baabaa	6
7 alabala	

P070303: Petrecerea nebunatică
La "petrecerea nebunatică" participă *n* persoane. Acestea sunt așezate în jurul unei mese. După fiecare minut o pereche de vecini își schimbă locurile între ei.

Sarcina voastră este să determinați timpul minim, exprimat în minute, după care participanții la petrecere stau în ordine inversă.

Date de intrare
Fișierul de intrare **C.IN** conține pe prima linie numărul de teste. În continuare, pe o linie separată, pentru fiecare test, se află numărul *n* de persoane care participă la "petrecerea nebunatică".

Date de ieșire
Rezultatele vor fi furnizate la ieșirea standard. Pentru fiecare test din fișierul de intrare se va scrie pe o singură linie numărul minim de minute după care participanții sunt așezați în ordine inversă. Rezultatele se vor scrie în ordinea în care testele apar în fișierul de intrare.

Restricție
• $1 \le N \le 32767$.

Exemplu

C.IN	*Ieșirea standard*
3	2
4	4
5	6
6	

P070304: Supermagazin
Un supermagazin are de vânzare o mulțime `Prod` de produse. Pentru fiecare produs `x` ∈ `Prod` vândut până la scurgerea a d_x unități întregi de timp, care sunt măsurate începând cu momentul începerii vânzării de produse, se obține un profit p_x. Vânzarea unui produs durează o unitate de timp.

O planificare a vânzărilor este o submulțime ordonată `Sell` ⊆ `Prod`, astfel încât oricare produs `x` ∈ `Sell` se vinde înainte de expirarea timpului d_x sau chiar în momentul d_x.

Profitul `Profit(Sell)` al unei planificări `Sell` a vânzărilor este egal cu suma profiturilor care se obțin din vânzarea tuturor produselor din planificare. O planificare optimă este o planificare pentru care profitul este maxim.

Să considerăm produsele `Prod = {a, b, c, d}` pentru care: $(p_a, d_a) = (50, 2)$, $(p_b, d_b) = (10, 1)$, $(p_c, d_c) = (20, 2)$, respectiv $(p_d, d_d) = (30, 1)$. Planificările și profiturile corespunzătoare care se pot obține pentru aceste produse sunt:

planificare	*profit*	*planificare*	*profit*
{a}	50	{a, c}	70
{b}	10	{c, a}	70
{c}	20	{b, c}	30
{d}	30	**{d, a}**	**80**
{b, a}	60	{d, c}	50

De exemplu, planificarea `Sell = {d, a}` ne indică faptul că vânzarea produsului `d` începe la momentul 0 și se termină la momentul 1, în timp ce vânzarea produsului `a` începe la momentul 1 și se încheie la momentul 2. Fiecare dintre cele două produse sunt vândute până la expirarea termenului limită. Această planificare este optimă și profitul său este 80.

Sarcina voastră este să scrieți un program care pentru o mulțime de produse determină profitul care se poate obține pentru o planificare optimă.

Date de intrare
Prima linie a fișierului de intrare **D.IN** este format din mai multe teste.

Fiecare test este format dintr-un număr natural *n*, care reprezintă numărul de produse din mulțime, urmat de *n* perechi (p_i, d_i), $1 \le i \le n$, care reprezintă profitul obținut în urma vinderii produsului p_i până la termenul limită d_i.

Testele apar în fișierul de intrare unele după altele. Datele din acest fișier pot fi separate între ele prin oricâte spații (incluzând sfârșitul de linie).

Datele de intrare se termină cu *EOF* și sunt corecte.

Date de ieșire
Rezultatele se vor scrie la ieșirea standard, astfel pentru fiecare test în parte se va scrie pe o linie separată profitul planificării optime. Rezultatele pentru teste se scriu în ordinea în care testele apar în fișierul de intrare.

Restricții
• $0 \le n \le 10.000$;
• $1 \le p_i \le 10.000$, $1 \le i \le n$;
• $1 \le d_i \le 10.000$, $1 \le i \le n$.

Exemplu

D.IN

```
4   50 2   10 1    20 2    30 1

7   20 1    2 1   10 3   100 2 8 2   5 20   50 10
```

Ieșirea standard

```
80
185
```

Explicație

Primul test este chiar exemplul prezentat în enunț, deci profitul planificării optime este 80.

Pentru cel de-al doilea test avem 7 produse. Planificarea optimă conține toate produsele din *P*, mai puțin cele cu indicii 2 și 5, deci profitul planificării optime pentru acest set de date este 185.

P070305: Stăpânul inelului

Frodo are de îndeplinit o misiune nobilă, dar dificilă. El trebuie să distrugă un inel magic și blestemat. În această aventură el trebuie să treacă printr-un loc periculos numit *Mordor* și să arunce inelul într-o prăpastie în flăcări.

Frodo a plecat de acasă de o bună bucată de timp și, momentan, merge pe un drum drept și lung, presărat cu tufișuri din loc în loc. Fiind foarte obosit, el consideră că ar fi timpul să se odihnească. Singurul loc sigur în care poate poposi este un tufiș a cărui poziție poate fi calculată folosind o formulă magică. Această formulă folosește valoarea lui *P* care reprezintă produsul distanțelor dintre perechile de tufișuri adiacente care se află de-a lungul drumului.

Frodo cunoaște numai distanțele dintre fiecare pereche de tufișuri și formula magică, dar, din păcate, nu cunoaște valoarea lui *P*. Puteți să-l ajutați să determine această valoare?

Date de intrare

Fișierul de intrare **E.IN** conține mai multe seturi de date.

Fiecare set de date este format dintr-un număr *n*, care reprezintă numărul de distanțe, urmat de *n* distanțe în ordine crescătoare.

Numerele din fișierul de intrare sunt separate prin oricâte spații (înclusând sfârșitul de linie).

Date de ieșire

Rezultatele se vor scrie la ieșirea standard. Pentru fiecare set de date trebuie scrisă pe o linie separată valoarea calculată pentru *P*, dacă aceasta se poate calcula sau textul "`No solution`" în caz contrar. Rezultatele pentru fiecare test se vor scrie în ordinea în care testele apar în fișierul de intrare.

Restricții și precizări

- numărul de tufișuri care se află de-a lungul drumului este cel puțin doi și cel mult 1.000;
- $1 \le P \le 1.000.000.000$.

Exemplu

E.IN

```
6
1 2   2    3   3 5

3
1 2 2
```

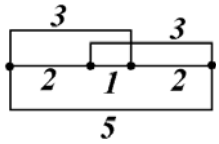


Figura 1

Ieșirea standard

```
4
No solution
```

Explicație

În figura 1 este ilustrată poziția tufișurilor pentru primul set de date.

Pentru cel de-al doilea set de date pozițiile tufișurilor care se află de-a lungul drumului nu pot fi deduse din valorile datelor de intrare, deci distanțele dintre punctele adiacente nu pot fi calculate.

P070306: Subsecvența comună

O subsecvență a unei secvențe date este chiar subsecvența dată cu mai puține elemente (posibil nici unul).

Fie o secvența $X = (x_1, x_2, ..., x_m)$. O secvență $Z = (z_1, z_2, ..., z_k)$ este subsecvență a lui X dacă există o secvență strict crescătoare $(i_1, i_2, ..., i_k)$ de indici din X, astfel încât pentru $1 \le j \le k$ să avem $x_{i_j} = z_j$. De exemplu, secvența $Z = (a, b, f, c)$ este o subsecvență a lui $X = (a, b, c, f, b, c)$ cu secvența de indici (1, 2, 4, 6).

Dându-se două secvențe X și Y, determinați valoarea lungimii celei mai lungi subsecvențe comune lui X și Y.

Date de intrare

Fișierului de intrare **F.IN** conține mai multe seturi de date. Datele pentru fiecare set constau în două șiruri de caractere care reprezintă cele două secvențe X și Y. Secvențele din fișierul de intrare sunt separate prin oricâte spații.

Date de ieșire

Rezultatele se vor scrie la ieșirea standard. Pentru fiecare set de date, pe o linie separată, se va scrie lungimea subsecvenței comune maxime. Rezultatele pentru fiecare set de date se vor scrie în ordinea în care apar în fișierul de intrare.

Exemplu

F.IN		*Ieșirea standard*
`abcfbc`	`abfcab`	`4`
`programming`	`contest`	`2`
`abcd`	`mnp`	`0`

P070307: Rețeaua electrică

O rețea electrică este formată din mai multe noduri (producători, distribuitori și consumatori) conectate între ele prin linii de transport.

Într-o rețea electrică există cel puțin un producător.

Un nod u poate fi aprovizionat cu o cantitate $s(u) \geq 0$ de curent electric, poate produce $0 \leq p(u) \leq p_{max}(u)$ curent electric, poate consuma $0 \leq c(u) \leq \min(s(u), c_{max}(u))$ curent și poate distribui $d(u) = s(u) + p(u) - c(u)$.

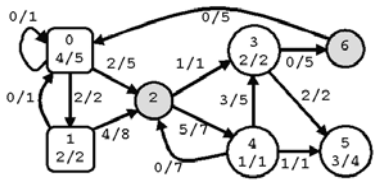
Se aplică următoarele restricții:
- $c(u) = 0$ pentru orice producător,
- $p(u) = 0$ pentru orice consumator și
- $c(u) = 0$ și $p(u) = 0$ pentru orice distribuitor.

Între două noduri u și v există cel mult o linie de transport. Aceasta poate transporta $0 \leq l(u, v) \leq l_{max}(u, v)$ curent electric de la nodul u la nodul v.

Fie $Con = \sum_u c(u)$ cantitatea de curent consumată în rețea. Problema cere să se determine cea mai mare valoare a lui *Con* care se poate atinge.

În figura următoare este prezentat un exemplu de rețea electrică.

u	*tip*	*s(u)*	*p(u)*	c(u)	*d(u)*
0	producător	0	4	0	4
1	producător	2	2	0	4
3	consumator	4	0	2	2
4	consumator	5	0	1	4
5	consumator	3	0	3	0
2	distribuitor	6	0	0	6
6	distribuitor	0	0	0	0



Eticheta *x/y* care apare în dreptul unui producător u indică faptul că $p(u) = x$ și $p_{max}(u) = y$. Eticheta *x/y* care apare în dreptul unui consumator u indică faptul că $c(u) = x$ și $c_{max}(u) = y$. Eticheta *x/y* care apare în dreptul unei linii de transport (u, v) indică faptul că $l(u, v) = x$ și $l_{max}(u, v) = y$. Puterea consumată de această rețea este *Con* = 6. Se observă că mai există și alte stări ale rețelei, dar valoarea *Con* nu poate depăși valoarea 6.

Date de intrare

Numele fișierului de intrare este **G.IN**. Acesta conține mai multe seturi de date. Fiecare set de date codifică o rețea electrică.

Un set de date începe cu patru numere naturale n, n_p, n_c, m care reprezintă numărul total de noduri, numărul de producători, numărul de consumatori, respectiv numărul de linii de transport. Cele patru numere sunt urmate de m triplete $(u,v)z$ care indică faptul că linia de transport care leagă nodurile u și v are capacitatea maximă $l_{max}(u, v)$ egală cu z. Nodurile sunt numerotate începând cu 0. Cele m triplete sunt urmate de n_p perechi $(u)z$ care reprezintă faptul că producătorul u poate produce $p_{max}(u) = z$ curent.

Setul de date se încheie cu n_c perechi $(u)z$ care reprezintă faptul că un consumator u poate consuma $c_{max}(u) = z$ curent electric.

Datele din fișierul de intrare sunt numere întregi și sunt separate prin oricâte spații în afară de tripletele și perechile, care nu conțin spații.

Datele de intrare sunt corecte și se termină cu *EOF*.

Date de ieșire

Rezultatele se vor scrie la ieșirea standard. Pentru fiecare set de date, pe o linie separată, se va scrie valoarea maximă obținută pentru *Con*.

Restricții

- $0 \leq n \leq 100$;
- $0 \leq n_p \leq n$;
- $1 \leq m \leq n^2$;
- $0 \leq c_{max}(u), p_{max}(u) \leq 10.000$, $0 \leq u < n$;
- $0 \leq l_{max}(u, v) \leq 10.000$, $0 \leq u, v < n$.

Exemplu

H.IN

```
2 1 1 2 (0,1)20 (1,0)10 (0)15 (1)20
7 2 3 13 (0,0)1 (0,1)2 (0,2)5 (1,0)1 (1,2)8
   (2,3)1 (2,4)7 (3,5)2 (3,6)5 (4,2)7 (4,3)5
   (4,5)1 (6,0)5 (0)5 (1)2 (3)2 (4)1 (5)4
```

Ieșirea standard

```
15
6
```

Explicație

Primul set de date codifică o rețea electrică cu două noduri și două linii de transport. Producător este nodul 0 și $p_{max}(0) = 15$, iar consumator este nodul 1 și $c_{max}(1) = 20$. Pentru cele două linii avem $l_{max}(0, 1) = 20$ și $l_{max}(1, 0) = 10$. Valoarea maximă pentru *Con* este 15.

Cel de-al doilea set de date codifică rețeaua din exemplul din enunț, deci valoarea maximă pentru *Con* este 6.

P070308: Pompierii

Un matematician își păstra toate documentele într-un dulap situat în apropierea biroului său. Într-o zi a izbucnit un incendiu care i-a distrus o parte din documente. Din fericire, câteva dintre ecuațiile pe care le-a rezolvat de-a lungul carierei sale erau doar parțial deteriorate. Fiecare ecuație era formată dintr-o expresie în partea stângă și rezultatul în partea dreaptă. Expresiile parțial deteriorate constau din toate numerele și parantezele, dar din păcate, unele dintre semnele aritmetice nu se mai puteau distinge din cauza focului. O altă problemă era că rezultatele ecuațiilor au fost împrăștiate și matematicianul nu era sigur dacă un anumit răspuns corespundea rezultatului unei anumite ecuații.

Sarcina voastră este să-l ajutați pe matematician să afle dacă expresiile și rezultatele pe care a reușit să le salveze se potrivesc.

Pentru a-l putea ajuta, aveți la dispoziție o expresie care conține numere întregi cuprinse între 1 și 999, semnele matematice simple (+, -, ·, /), parantezele și semnul de întrebare care înlocuiește semnele pierdute.

Pentru fiecare expresie dată singura voastră sarcină este să spuneți dacă o expresie poate sau nu poate avea rezultatul cerut.

Pentru a vă ajuta, matematicianul a ales numai expresii care au următoarele restricții:
- expresiile nu conțin mai mult de 100 de simboluri;
- parantezele nu conțin decât cel mult o operație matematică și doi operanzi, dar oricare dintre cei doi operanzi sau amândoi pot avea forma unei expresii între paranteze;
- constantele prezente în expresii nu au semn (expresiile nu conțin numere negative);
- numărul maxim de semne de întrebare din expresii este mai mic sau egal cu 10.

Calcularea rezultatului ar trebui să se facă după următoarele reguli:
- operatorii · și / au prioritate mai mare decât + și -, iar parantezele pot schimba prioritatea;
- operatorii +, -, · și / sunt asociativi la stânga, ceea ce înseamnă că se grupează de la stânga la dreapta, astfel, dacă a, b și c sunt numere avem: $a \cdot b \cdot c = (a \cdot b) \cdot c$, $a / b / c = (a / b) / c$, $a / b \cdot c = (a / b) \cdot c$, $a + b + c = (a + b) + c$, $a - b + c = (a - b) + c$ etc.
- când se împart două numere întregi, trebuie să ignorați partea fracțională, de exemplu 2 / 5 = 0, 9 / 5 = 1, 100 / 6 = 16.

Date de intrare

Prima linie a fișierului de intrare **H.IN** conține un singur număr N care reprezintă numărul de ecuații prezente în fișier. Următoarele $2 \cdot N$ linii conțin ecuațiile.

O ecuație este definită pe două linii. Prima linie conține partea stângă a ecuației, iar cea de-a doua rezultatul (partea dreaptă a ecuației) dat sub forma unui număr întreg.

Fișierul de intrare nu conține spații, iar expresiile ecuațiilor nu au erori de sintaxă.

Date de ieșire

Rezultatele se vor scrie la ieșirea standard. Pentru fiecare ecuație, pe o linie separată, se va scrie cuvântul "yes" dacă ecuația poate avea valoarea cerută și cuvântul "no" în caz contrar.

Exemplu

H.IN
```
3
1?((2*(3*4))+(5+6))
35
1?2*3+4-14
0
1?3*4/5*6+12
11
```

Ieșirea standard
```
yes
no
no
```

P070309: Polinoame binare

O funcție f definită pe mulțimea $\{0, 1\}^n$ de vectori binari și care ia valori din mulțimea $\{0, 1\}$ poartă numele de *funcție booleană de n variabile* și se notează cu $f(x_n, x_{n-1}, ..., x_1)$. Pentru criptografie sunt importante câteva proprietăți ale funcțiilor *booleene*.

Notăm cu $B(n, k)$ mulțimea vectorilor binari n-dimensionali care au k valori egale cu 1.

Pentru o funcție dată f se cere să se determine numărul de vectori de forma $(b_n, b_{n-1}, ..., b_1)$ din $B(n, k)$ pentru care $f(b_n, b_{n-1}, ..., b_1) = 1$.

O funcție *booleană* este unic determinată de valorile polinomului asociat modulo 2.

În aceste polinoame se folosesc doar operațiile de adunare și înmulțire modulo 2, după cum indică tabelul 1.

În polinomul asociat unei funcții orice produs de m variabile $x_{i1}, x_{i2}, ..., x_{im}$ poate să participe sau poate să nu participe. În general, forma acestor polinoame este $a_0 + a_1x_1 + a_2x_2 + a_3x_2x_1 + a_4x_3 + a_5x_3x_1 + ... + a_Nx_nx_{n-1}...x_1$, unde coeficienții a_j, $1 \le j \le N = 2^{n-1}$, pot fi 0 sau 1, iar dacă sunt 0, atunci produsele respective pot fi ignorate.

De exemplu, funcția *booleană* de disjuncție reprezentată în tabelul 2 este $0 + 1 \cdot x_1 + 1 \cdot x_2 + 1 \cdot x_1 \cdot x_2 = x_1 + x_2 + x_1 \cdot x_2$.

+	0	1
0	0	1
1	1	1

*	0	1
0	0	0
1	0	1

Tabelul 1

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

Tabelul 2

Date de intrare

Fișierul de intrare este **F.IN** conține pe prima linie numărul T al seturilor de date. Pe fiecare dintre următoarele T linii se află câte un set de date.

Un set de date este format din două numere n și k separate printr-un singur spațiu. Aceste numere sunt urmate de un șir de 2^n cifre 0 și 1 care reprezintă coeficienții polinomului.

Date de ieșire

Rezultatele se vor scrie la ieșirea standard. Pentru fiecare set de date, pe o linie separată, se va scrie numărul de vectori determinați. Rezultatele pentru fiecare set se vor scrie în ordinea în care apar în fișierul de intrare.

Exemplu

I.IN
```
3
2 1 0111
4 2 1000000000000000
5 3 00000000000000000000000000000001
```

Ieșirea standard
```
2
6
0
```