CH01. Cros (50 puncte)

La crosul anual al orasului Tg Mures, s-au inscris la linia de plecare o multime de sportivi. Sunt sportivi de performanta sau nu, persoane tinere si mai putin tinere, veniti din toata tara si chiar din strainatate. Strainii, atrasi aici de mitul orasului bilingv, descopera un oras mic, frumos, primitor.

Startul se da de la intersectia de la catedrala, iar sosirea este in intersectia de strazi din cartierul Tudor. Ca sa nu se creeze invalmaseala, multimea de alergatori va fi dirijata pe mai multe strazi, care au in comun faptul ca fac parte din drumuri de lungime minima dintre punctul de start si cel de sosire. Pentru simplificare, se considera ca toate strazile pe care se alearga, au aceeasi lungime. Intersectiile au fost etichetate cu numere intre 1 si n, iar strazile dintre ele pot fi circulate in ambele sensuri..

Pentru dirijarea fluxului de concurenti, Primaria va amplasa puncte de control si de prim ajutor , in toate intersectiile, care se gasesc pe drumuri minime intre plecare si sosire.

In acest moment, este nevoie de tine. Ajuta Primaria sa identifice aceste puncte.

Fisierul cros.in contine pe prima linie 3 numere intregi n si S, si F, reprezentand numarul de intersectii, punctul de start si cel de finis. Pe urmatoarele linii, despartite prin spatiu, cate doua numere, reprezentand etichetele intersectiilor intre care exista strada directa.

Fisierul cros.out va contine pe o singura linie, despartite prin spatiu, toate etichetele intersectiilor de strazi, in ordine crescatoare.

Exemple:

CROS.IN	CROS.OUT
4 1 4 1 2 1 3 2 4 3 4	1 2 3 4

CROS.IN	CROS.OUT
5 2 5 1 2 2 3 3 4 4 5 3 5 5 1	1 2 3 5

Restrictii:

1 <= N, S, F <= 1000

Timp maxim de executie/test: 1 secunda

CH02. Cube zero(50 puncte)

Clod tocmai a terminat de vazut filmul "Cube Zero" care l-a impresionat destul de tare. Povestea acestui film era bazata pe ideea inchiderii unor oameni intr-o cladire in forma de cub care era formata din camere in forma de cub ce pot comunica una cu cealalta daca sunt vecine. Problema era ca oamenii inchisi in acel cub trebuiau sa iasa afara, dar unele camere contineau capcane si o traversare a unei camere ce contine o capcana s-ar fi sfarsit tragic. Clod este curios pentru o camera initiala in care este pus un om si o configuratie data a capcanelor care este drumul cel mai scurt de iesire din

cub care nu trece prin nici o camera cu capcane. Din cub se poate iesi numai prin camera de coordonate 1,1,1. Cubul este reprezentat de o matrice tridimensionala de dimensiuni N*N*N, o camera de coordonate (x,y,z) contine capcane daca in matrice elementul a[x][y][z] este egal cu 1 si nu contine capcane daca elementul este egal cu 0. O persoana poate trece numai in camerele vecine lateral sau orizontal cu camera curenta, nu se poate misca in diagonala.

Intrare:

cube.in

N // dimensiunea cubului

X Y Z // cele trei coordonate ale camerei initiale

Pe liniile urmatoare se vor afla N blocuri fiecare bloc reprezentand o matrice de N*N elemente

al i-lea bloc va contine camerele pentru care x e egal cu i

Fiecare bloc este despartit de urmatorul bloc printr-o linie goala.

A j-a linie dintr-un bloc i reprezinta camerele cu x egal cu i si y egal cu j, separate printr-un spatiu.

Al k-lea element dintr-a j-a linie dintr-al i-lea bloc reprezinta camera de coordonate (i,j,k). La iesire se va afisa numarul minim de pasi pentru a ajunge din camera initiala in camera (1,1,1) sau -1 daca nu exista solutie.

Restrictii: 1 <= N <= 20

Exemplu:

CUBE.IN	**CUBE.OUT**
3 3 3 3 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0	6

CH03. Partid (50 puncte)

Gica Becalescu a infiintat un nou partid, Partidul Nepasatorilor, cu care vrea sa castige alegerile din Tg Mures. Evident primii inscrisi au fost rudele si prietenii sai. Gica trebuie sa organizeze partidul pe nivele de raspundere. Astfel el a stabilit niste relatii de sef-subordonat, in functie de obligatiile pe care le avea fata de anumite persoane. Acum el urmeaza sa aseze membrii partidului pe nivele, tinand cont ca pe un anumit nivel nu pot fi 2 persoane intre care exista relatie de sef-subordonat.

Sa se determine ierarhia noului partid.

Prima linie a fisierului de intrare partid.in va contine numarul n al persoanelor si numarul m al relatiilor de subordonare, separate printr-un spatiu. Fiecare dintre urmatoarele m linii va descrie o relatie de subordonare; ea va contine doua numere x si y, separate printr-un spatiu, cu semnificatia: persoana y este subordonata persoanei x.

Fisierul de iesire partid.out va contine k linii, k fiind numarul de nivele. Fiecare din aceste linii va contine persoanele care alcatuiesc un nivel, in ordinea crescatoare a numarului de ordine. Pe prima linie se afla persoanele care nu au nici o relatie de subordonare fata de alta persoana.

Restrictii si precizari

0<= n<= 100

n<=m<=1000

pot exista persoane care sa aiba relatie de subordonare fata de mai multe persoane

Exemple

partid.in	partid.out
7 7 7 3 7 1 2 1 2 4 2 5 1 4 3 6	2 7 1 3 5 4 6

Timp maxim de executie/test: 1 secunda

CH04. Concerte (80 puncte)

A inceput festivalul "Zilele Colegiului National Al. Papiu Ilarian ". Anul acesta s-au amenajat scene si estrade in aer liber in numar mai mare decat de obicei, pentru ca sunt invitate sa sustina concerte, multe formatii de muzica. De data aceasta, Vasile, este decis, mai mult ca oricand, sa nu piarda nici un concert important. Din pacate, intrarea la spectacole este cu plata. Biletele se vand in avans, astfel, ca se cunoaste ora de inceput, ora de sfarsit si pretul fiecarui concert. Vasile este pus in situatia neplacuta sa renunte la unele dintre ele, intrucat intervalele de timp se suprapun in unele cazuri si MAI GRAV, suma de bani pe care o are este limitata. Speranta lui Vasile esti tu, prietenul lui. Ajuta-l sa aleaga un numar cat mai mare de spectacole pe care le poate vedea in intregime.

Fisierul concerte.in contine pe prima linie 2 numere intregi n si S, reprezentand numarul de concerte si suma de bani pe care o are Vasile. Pe urmatoarele n linii, cate 3 numere intregi. Numerele de pe linia inia i + 1, reprezinta ora de inceput, ora de sfarsit si pretul concertului i.

Fisierul concerte.out contine pe o singura linie, doua numere intregi, nmax si smin. Primul numar reprezinta numarul maxim de spectacole care pot fi vazute integral iar smin este costul lor. Pot fi alese doua concerte la care coincid ora de sfarsit a primului cu ora de inceput al celuilalt. Se pot alege de asemenea si spectacole la care ora de inceput coincide cu cea de sfarsit . Daca exista mai multe variante cu numar maxim de spectacole, trebuie sa alegi varianta cea mai ieftina pentru Vasile.

Restricții

1 <= n <= 100

1<= S <= 10000

Pentru orice spectacol, hs <= hf unde hs, hf - orele de inceput si de sfarsit

Exemple :

CONCERTE.IN	CONCERTE.OUT
4 15 2 3 2 3 6 3 5 7 4 8 9 5	3 10
CONCERTE.IN	**CONCERTE.OUT**
3 5 3 4 4 5 6 3 1 2 2	2 5

Timp de execuţie/test: 1 secundă/test

CH05. Bastion (60 puncte)

Amenintzat de invazia trupelor romane, Decebal a construit un numar de bastioane care sa ii apere capitala, Sarmisegetuza. Pentru a economisi bani, capetenia insarcinata de Decebal sa realizeze lucrarea, a decis sa nu construiasca cai de acces directe intre oricare 2 bastioane. De la un bastion la alt bastion intre care exista cale de acces directa trupele se pot deplasa intr-o ora. Cand s-au terminat lucrarile, Decebal a descoperit ca unele bastioane sunt greu de sustinut in cazul unui atac deoarece se ajunge intr-un timp mai mare la ele. Asha ca l-a pus pe Zaris, capetenia responsabila cu constructia, sa-i dea o lista cu cele mai indepartate bastioane, precum si cu timpul necesar pentru a ajunge din capitala la acele bastioane.

Sa se determine bastioanele aflate la cea mai mare distanta fata de capitala Sarmisegetuza, precum si timpul necesar pentru a ajunge aici.

Prima linie a fisierului de intrare bastion.in va contine numarul n al bastioanelor, cu precizarea ca 1 este numarul care reprezinta Sarmisegetuza si numarul m al cailor de acces construite intre bastioane, separate printr-un spatiu. Fiecare dintre urmatoarele m linii va descrie o cale de acces; ea va contine doua numere x si y, separate printr-un spatiu, cu semnificatia: exista cale de acces directa intre bastionul x si bastionul y.

Fisierul de iesire bastion.out va contine pe prima linie un numar t, care reprezinta timpul in care se ajunge la cele mai indepartate bastioane, iar pe linia a doua aceste bastioane in ordine crescatoare.

Restrictii si precizari

- 0<= N<= 100
- N<=M<=1000
- nu pot exista doua sau mai multe cai de acces intre aceleasi doua bastioane

Exemple

BASTION.IN	BASTION.OUT
6 11 1 2 1 3 1 5 2 3 2 4 2 5 2 6 3 4 3 5 4 5 5 6	2 4 6

Timp maxim de executie/test: 1 secunda

CH06. Reduceri(80 puncte)

Consideram un sir de N numere naturale distincte x1, x2, ... xN. Vom numi reducere de lungime p eliminarea de la unul dintre cele doua capete ale sirului a secventei formate din primele, respectiv ultimele p elemente din sir. Mai exact, o reducere de lungime p consta fie din eliminarea elementelor x1, x2, ...,xp, fie eliminarea elementelor xN-p+1, xN-p+2, ..., xN.

Evident, dupa reducere, sirul devine mai scurt cu p elemente. Sirului obtinut in urma reducerii i se pot aplica in continuare operatii de reducere de diferite lungimi, procedeul continuand pana cand sir devine vid. Costul unei reduceri se calculeaza ca fiind diferenta absoluta dintre elementele de la capetele secventei care se elimina inmultita cu numarul de elemente ale secventei. Presupunand ca reducerea este formata din secventa xi, xi+1, ..., xk, atunci costul reducerii este |xi – xk|*(i-k+1). Daca reducerea este de lungime 1, atunci costul sau este dat de valoarea elementului eliminat. Costul total al reducerilor este dat de suma tuturor reducerilor aplicate sirului.

Sa se determine costul total maxim al reducerilor aplicate sirului.

Fisierul de intrare reduceri.in contine pe prima linie numarul natural N, iar pe urmatoarea linie se afla N numere naturale distincte separate prin cate un spatiu, reprezentand sirul.

Fisierul reduceri.out va contine o singura linie pe care va fi scris costul total maxim care se poate obtine aplicand reduceri succesive sirului.

Restrictii si precizari : 3<=N<=100 Elementele sirului sunt numere naturale distincte si nenule din intervalul 1..1000. Poate fi aplicata sirului o singura reducere, care consta in eliminarea intregului sir.

Exemplu

REDUCERI.IN	REDUCERI.OUT	Explicatii
6 54 29 196 21 133 118	768	Se vor efectua 3 reduceri. Prima reducere: se elimina primele 3 elemente cu un cost de 426. A doua reducere: se elimina ultimul element, cu un cost de 118. Ultima reducere: se elimina ce a ramas din sir, adica 21 si 133, cu un cost de 224. Costul total este 426+118+224=768

Timp maxim de executie/test: 1 secunda

CH07. asm (100 puncte)

Limbajul de asamblare lucreaza cu 4 registrii de 4 biti **ax,bx,cx,dx** pe care poate aplica urmatoarele operatii:

Operatie	**Rezultat**
MOV a,b	a = b
ADD a,b	a = a + b

SUB a,b	a = a - b //numerele nu au semn!!!
	// 1 - 3 = 14 !

MUL a,b	a = a * b [ultimi 4 biti…]
DIV a,b	a = a / b [catul impartiri lui a la b]
MOD a,b	a = a % b [a devine restul impartirii lui a la b]
NOT a	a = ~a [se neaga a pe biti]
AND a,b	a = a & b
OR a,b	a = a | b
XOR a,b	a = a ^ b

Se cunoaste starea initiala a registrilor si starea lor finala dupa efectuarea unor linii de cod, tu trebuie sa afli numarul minim de linii de cod care ar obtine acelasi rezultat

Fisierul de intrare **asm.in** contine pe prima linie t numarul de teste apoi urmeaza t linii, fiecare linie continand 8 numere **ax,bx,cx,dx,ax`,bx`,cx`** si **dx`** reprezentand valoirile initiale si cele finale ale registrilor.

Fisierul de iesire **asm.out** contine **t** linii fiecare continand numarul minim de operatii care trebuie facut pentru a obtine acel rezultat

Exemplu

ASM.IN	**ASM.OUT**
3 1 1 1 1 2 2 2 2 1 1 2 2 2 2 1 1 1 1 1 1 0 0 0 1	4 4 3

Explicatie: la primul test se poate aplica urmatorul cod

```
ADD ax, bx
ADD ax, cx
ADD ax, dx
ADD ax, ax
```

La al doilea test codul

```
MOV bx, cx
MOV dx, ax
MOV ax, bx
MOV cx, dx
```

Si la al treilea

```
SUB ax, dx
SUB bx, dx
SUB cx, dx
```

CH08. Supersets (100 puncte)

S-a obsetvat ca calculatoarele obisnuite nu pot sa rezolve toate problemele de natura algoritmica asa ca Xcsi si Take s-au apucat sa experimenteze cu conceptul de SuperSet Stack Computer, care este un calculator care rezolva foarte rapid operatii pe o stiva de multimi.

Cand se porneste un SuperSet Stack Computer stiva de multimi este goala, si pe ea se pot aplcia urmatoarele operatii:

- **PUSH** – se pune in varful stivei multimea vida "{}"
- **POP** - se scoate multimea din varful stivei
- **DUPE** – Se pune in varful stivei o copie a multimi care se afla in varful stivei
- **UNION** – Se scoate de doua ori cea mai de sus multime din stiva si se adauga reuniunea celor doua multimi in varful stivei
- **INTERSECT** - Se scoate de doua ori cea mai de sus multime din stiva si se adauga intersectia celor doua multimi in varful stivei
- **ADD** - Se scoate de doua ori cea mai de sus multime din stiva pe care se notam A si B se adauga in varful stivei multimea {A,B}
- Operatile **UNION**, **INTERSECT** si **ADD** nu se pot efecta daca stiva contine mai putin de doua multimi
- Operatile **POP** si **DUPE** se pot efectual doar daca stiva nu este goala

Fisierul de intrare **superset.in** contine pe prima linie N numarul de operatii care trebuie efectuate, urmeaza N linii care contin operatile care trebuie efectuate.

Fisierul de iesire **superset.out** contine o linie pentru fiecare multime din stiva pe care se va afisa multimea. Multimile se vor afisa de la varf in jos.

Restrictii

- Operatile din fisierul de intrare se pot efectua
- Xcsi si Take nu exista in realitate si daca ar exista ar fi niste lenesi si jumatate
- Multimile care contin aceleasi elemnte sunt echivalente [ex. {{},{{}}} si {{{}},{}}] asa ca se pot afisa oricum in fisierul de iesire

Exemplu

SUPERSET.IN	**SUPERSET.OUT**
PUSH PUSH PUSH ADD DUPE DUPE ADD PUSH PUSH PUSH ADD ADD DUPE DUPE PUSH ADD INTERSECT	{{}} {{{},{}},{}} {{{},{}},{{},{}}} {{},{}} {}

CH09. Sirp (100 puncte)

Fie un sir de **n** numere naturale strict pozitive $a_1, a_2, \ldots, a_n$ pe care se pot aplica doua operatii PLUS(k) care creste valoarea lui a_k cu 1 si MINUS(k) care scade valoarea lui a_k cu 1. Sa se determine numarul minim de operatii care trebuie facute astfel incat toate elementele sirului sa aiba aceasi valoare

Fisierul de intrare ***sir.in*** contine pe prima linie numarul **t** de teste dupa care urmeaza 2***t** linii care reprezinta testele. Pentru fiecare test pe prima linie se afla **n** numarul de numere ale sirului iar pe a doua linie se afla sirul de numerele

Fisierul de iesire ***sir.out*** va contine **t** linii pe care se afla raspunsurile la teste.

Restrictii

- **n** < 1.000.000
- 90% vor avea **n** < 1000
- numerele sirului se pot reprezenta pe 32 biti cu semn

Exemplu

SIRP.IN	**SIRP.OUT**
4 4 1 1 1 1 4 1 2 3 4 4 8 9 1 3 4 100 200 400 800	0 3 12 900

CH10. Poli (100 puncte)

Odrog e cam varza cu scoala … la tema de mate trebuie sa gaseasca radacinile intregi a **T** polinoame de grad **N**. Hai … fi de treaba si ajutal pe grasutz.

Fisierul de intrare **poli.in** contine pe prima linie numarul **T** de teste apoi urmeaza 2***T** linii cate 2 linii pentru fiecare test. Pentru un test pe prima linie se afla numarul **N** reprezentand gradul polinomului iar pe urmatoarea linie se afla **N**+1 valori intregi reprezentand coeficienti polinomului $a_{\mathbf{N}},a_{\mathbf{N}-1},\ldots,a_1,a_0$.

Fisierul de iesire **poli.out** contine **T** linii de forma :

Test #x : nr_sol sol_1 sol_2 … sol_{nr_sol}

Unde x este numarul testului, nr_sol este numarul de radacini pe care le are acel polinom iar sol_1 , sol_2 … sol_{nr_sol} este sirul radacinilor ordonate crescator.

Restrictii

- coeficienti polinomului se pot reprezenta pe 16 biti cu semn
- gradul unui polinom este cel mult 128 si cel putin 1

Exemplu

poli.in	**poli.out**
3 1 1 2 2 1 2 -3 3 3 4 1 0	Test #1 : 1 -2 Test #2 : 2 -3 1 Test #3 : 2 -1 0

CH11. Back (100 puncte)

"Darkonienii fiind mult mai evoluati decat oamenii obisnuiti folosesc metode de calcul foarte rapide mult mai rapide decat cele ale unui om normal" a citit Raxvzan intr-o revista .

Neputand sa creada asa ceva el isi pune creierul la incercare si incearca sa gaseasca o metoda eficienta din punct de vedere al timpului

de a calcula intr-o jumatate de secunda numarul de posibilitati distincte de a-l

descompune pe N ca suma de puteri ale lui 2 folosindu-se de calculator dar deoarece era cam vãrzar a cerut ajutorul vostru .
Faceti un program ce calculeaza pentru Raxvan numarul de posibilitati distincte de a-l scrie pe N sub forma de suma de puteri ale lui 2 modulo 1000000000 (1 miliard).
Restrictii:1 <= N <= 4 000 000
Fisierul de intrare "back.in" contine numarul N.
Fisierul de iesire "back.out" va contine pe prima linie rezulatul modulo 1 000 000 000 obtinut . (modulo = %)
Exemplu:

BACK.IN	BACK.OUT
5	4

Explicatie:
```
5 = 1 + 1 + 1 + 1 + 1
5 = 1 + 1 + 1 + 2
5 = 1 + 2 + 2
5 = 1 + 4
```
Timp maxim de executare: 1 secundă/test.

CH12. Permutari (100 puncte)

Se consideră o permutare de dimensiune n. Sa se ridice la puterea m.

Prima linie din fisierul permutari.in va contine un numar n, reprezentand lungimea permutarii. Pe urmatoarea linie se afla n numere reprezentand elementele permutarii. Pe a treia linie se afla numarul m.

Fisierul de iesire va contine o singura linie pe care se vor afla n numere reprezentand permutarea cautata.

Restricţii $1 \le n \le 1\,000$

$1 \le m \le 1\,000\,000\,000$

EXEMPLUL 1	
PERMUT.IN	**PERMUT.OUT**
5 3 1 2 5 4 2009	2 3 1 5 4

Explicaţie :

Timp de execuţie/test: 1 secundă/test

CH13. SIR2 (100P)

Fie un sir de **N** elemente pe care se pot face doua tipuri de operatii:

U **x y** – elementul de pe pozitia **x** devine **y**

Q **x** – trebuie afisat cel mai mare i astfel incat suma primelor i elemente sa fie mai mica sau egala cu **x**

Fiind dat sirul initial simuleaza un set de operatii pe el.

Fisierul de intrare **sir2.in** contine pe prima linie numarul **N** reprezentand numarul de elemente ale sirului, pe a doua linie se afla **N** numere reprezentand sirul, pe linia a treia se afla numarul **M** reprezentand numarul de operatii care trebuie efectuate, in continuare se afla M linii reprezentand operatile in ordinea in care trebuie facute.

Fisierul de iesire **sir2.out** contine raspunsurile la operatile de tip Q.

Restrictii

- Sirul are cel mult 100.000 elemente
- Toate operatile se pot efectua pe 32 de biti fara semn

Exemplu

SIR2.IN	SIR2.OUT
3 1 2 3 5 Q 4 Q 10 U 3 1 U 2 1 Q 3	2 3 3

CH14. jobs (100p)

La un targ de joburi au participat **N** studenti care puteau sa isi depuna CV-ul la oricare joburi din cele **M** disponibile. Stiind fiecare student pentru care joburi si-a depus CV-ul determina numarul maxim de studenti care se pot angaja dupa aces targ.

Fisierul de intrare **jobs.in** contine pe prima linie T numarul de teste, urmeaza descrierea celor T teste. Pentru fiecare test prima linie contine numerele **N** si **M** cu semnificatia din enunt, apoi urmeaza **N** linii de forma nr_i opt_1 opt_2 … opt_{nri} unde nr_i reprezinta numarul de CV-uri depuse de studentul i iar opt_1 opt_2 … opt_{nri} reprezinta joburile la care si-a depus CV-ul.

Fisierul de iesire **jobs.out** contine pentru fiecare test o linie care contine numarul maxim de studenti care se pot angaja.

Restrictii

- **N,M** <= 10000
- Numarul de CV-uri depuse este cel mult 200.000
- Nimeni nu va angaja puturosi ca Xcsi si Take !

Exemplu

JOBS.IN	JOBS.OUT
2 1 2 1 1 3 3 3 1 1 2 2 3 3 2 3 1	1 3

PAP1.match (100 puncte)

Fie doua siruri de caractere **A** si **B**, determinati lungimea minima a unei subsecvente de a lui **B** care sa se potriveasca cu A.

- **B** este format din caractere mici ale alfabetului latin [‘a’,‘z’]
- **A** este format din caratere mici ale alfabetului latin [‘a’,‘z’], ‘?’ si ‘*’;
- literele se potrivesc intre ele ‘a’ cu ‘a’ , ‘b’ cu ‘b’ …
- ‘?’ se potriveste cu orice litera
- ‘*’ se potriveste cu orice secventa de litere [ex. “a”,“bwq”,“” etc.]

Input

In fisierul de intrare *match.in* pe prima linie se afla numarul **t** de teste apoi fisierul de intrare contine **t** perechi de linii reprezentand un test. Pentru fiecare test pe prima linie se afla sirul **A** si pe a doua linie se afla sirul **B**.

Output

In fisierul de iesire *match.out* pentru fiecare test se va afla o linie care contine un numar reprerentand lungimea minima a unei subsecvente de a lui **B** care se potriveste cu **A** sau mesajul "imposibil" in caz ca nu exista o astfel de secventa.

Restrictii

- **A** si **B** au lungimea cel mult 200
- **T** < 5555

Exemplu

MATCH.IN	MATCH.OUT
2 a*a*a babacccbaba a*b*c aaaabbbbbaaaaababababababab	8 imposibil

PAP2. medals (100 puncte)

Cele mai multe concursuri isi premiaza participantii cu medalii, deobicei de aur argint si bronz. Pentru ca romania nu are asa performante spectaculoase la fotbal s-a hotarat sa faca clasamentul medalilor folosind urmatoarea tehnica:

- se considera (a,b,c) ca find un triplet de numere oricare a,b,c numere reale
- produsul a doua triplete (a,b,c) si (x,y,z) este un numar real care este egal cu a*x + b*y + c*z
- se ia un triplet de forma ($1/10^i$, $1/10^j$, $1/10^k$)
- punctajul fiecarei tari este produsul dintre tripletul ales si (AUR,ARGINT,BRONZ) unde AUR este numarul de medalii de aur pe care le are tara, ARGINT este numarul de medalii de argint pe care le are tara si BRONZ este numarul de medalii de bronz pe care le are tara.
- tarile se pun in clasament in ordinea acestui punctaj si in caz de egalitate se pun alfabetic

Find date numarul de medalii pe care le are fiecare tara afla daca se poate alege un triplet de forma ($1/10^i$, $1/10^j$, $1/10^k$) astfel incat Romania sa fie campiona mondiala la fotbal.

Input

Fisierul de intrare *medals.in* contine pe prima linie numarul **t** de teste din fisierul de intrare. Pentru fiecare test pe prima linie se afla numarul **N** reprezentand numarul de tarii, iar pe urmatoarele **N** linii **Nume Aur Argint Bronz –** reprezentand numele tarii numarul de medali de aur,argint si respectiv bronz pe care le-a obtinut acea tara.

Output

Fisierul de iesire *medals.out* contine **t** linii care contin mesajul "DA" daca exista un triplet de forma ($1/10^i$, $1/10^j$, $1/10^k$) astfel incat Romania sa fie campiona mondiala la fotbal sau mesajul "NU" altfel.

Restrictii

- Numele tariilor este format din caractere mici si mari ale alfabetului latin ['A'..'Z','a'..'z']
- -1000 <= **i**,**j**,**k** <= 1000
- O tara nu poate obtine mai mult de 999 de medalii
- Odrog este unitate de masura pentru imensitate

Exemplu

MEDALS.IN	**MEDALS.OUT**
2 3 Romania 1 2 3 SUA 3 2 1 Franta 10 5 1 2 Romania 1 2 1 Italia 2 3 2	DA NU

KB01 Grădiniţa (100 puncte)

Într-o grădiniţă sunt **N** băieţi şi **N** fete, N≤250. Băieţii şi fetele mai dansează câte-o horă de ziua mamei. Fiecare băiat acceptă ca vecină de dans oricare fată şi numai fată la stanga si la dreapta, şi invers: fetele dansează fără nicio problemă cu oricare băiat, dar numai cu cate-un băiat la stanga şi la dreapta. Lămuriţi-o pe doamna educatoare, în câte moduri se pot aşeza copiii pentru o horă. Datele de intrare, un număr natural **N** de ex.numărul fetelor, se va citi din **HORA.IN**, iar datele de ieşire, numărul total al horelor posibile se va scrie în **HORA.OUT**.

Exemplu:

HORA.IN	**HORA.OUT**
3	6

Timp de execuţie/test: 1 secundă/test

KB02 Etaje colorate (50 puncte)

Un bloc are un număr de **N** etaje şi iniţial este vopsit în alb. Dorim să zugrăvim câteva etaje în roşu, şi dorim deasemenea ca două etaje consecutive să *nu* aibă aceeaşi culoare roşie. O " *scenografie*" de vopsire se codifică cu un şir format din cifre 0 şi 1, de lungime **N+1**, unde 1 reprezintă etajul vopsit în roşu, iar 0 etajul vopsit în alb. Redactaţi un program, care determină numărul de colorări diferite a clădirii, precum şi colorarea de ordinul k, în cazul în care colorările sunt ordonate lexicografic.

Datele de intrare,două numere naturale **N** şi **K**, 0≤**N**≤40, şi 1≤**K**≤100000000 se citesc din **etaj.in**, datele de ieşire se înscriu în fişierul text **etaj.out**. Fişerul de ieşire are următoarea structură: pe primul rând avem numărul total de colorări, iar pe al doilea rând avem colorarea cu numărul de ordine K, un şir format din cifrele 0 şi 1 separate prin câte-un spaţiu.

Exemplu:

ETAJ.IN	**ETAJ.OUT**
3 4	8 0 1 0 0

KB03 Segmente (80 puncte)

Fiind date un număr **N** (număr natural par, **N**>3, **N** ≤1000000) puncte în plan a.î. oricare trei puncte nu sunt coliniare, determinaţi numărul maxim de segmente de dreaptă care se pot construi cu extremităţile în aceste puncte astfel încât să nu se formeze nici un triunghi. Datele de intrare, **N** perechi de numere reale reprezentând **N** puncte cu proprietatea de mai sus, se vor citi din **TRI.IN**, iar numărul căutat se va scrie în **TRI.OUT**.
Exemplu:

TRI.IN	**TRI.OUT**
0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0	4

Timp de execuţie/test: 1 secundă/test

KB04 Bănci comerciale şi drumuri – episodul I (100 puncte)

Într-un oraş, fie acest oraş **X**, se raționalizează transportul local, în sensul că se vor rearanja drumurile astfel încât de la o bancă comercială **K** se poate ajunge la o altă bancă comercială **Q** numai printr-un singur drum. Edilul şef vrea să examineze fiecare variant, el ştie de ce…Ajutaţi edilul şef, calculând numărul de variante. Datele de intrare, un număr natural N≤ 300, reprezentând numărul bancilor comerciale din oraşul respectiv, se citesc din **BANCI1.IN**, iar datele de ieşire, numărul de variante se scriu în **BANCI1.OUT.**

Exemplu:

BANCI1.IN	**BANCI1.OUT**
3	3

Timp de execuţie/test: 1 secundă/test

KB05 Oraşul în trei culori (70 puncte)

În oraşul **Z** locuitorii s-au decis să-şi coloreze casele. În magazinele de vopsele se găsesc doar trei culori. Şi mai e o problemă: vecinii doresc să fie coloraţi în culori diferite. Verificaţi dacă se poate realiza dorinţa orăşenilor de a revopsi casele. Datele de intrare, numărul caselor şi legăturile de vecinătate, se citesc din **CULORI1.IN**, datele de ieşire, şirul de caractere **DA** sau **NU**, se va scrie în **CULORI1.OUT**. **Pentru rezolvarea problemei nu se va folosi generatorul de numere aleatoare, şi nu se vor scrie programe care afişează doar DA sau doar NU!**

Exemplu:

CULORI1.IN	**CULORI1.OUT**
3 1 2 2 3	DA

Timp de execuţie/test: 1 secundă/test

KB06 Numărul de cifre (100 puncte)

Din fişierul **NRCIF.IN** se citesc două numere natural **N**≤ 12345678şi **P**≤ 12345678 Calculaţi numărul de cifre al numărului $\mathbf{N^P}$. Datele de ieşire, numărul de cifre, se scrie în **NRCIF.OUT**

Exemplu:

NRCIF.IN	**NRCIF.OUT**
2 1	1

Timp de execuţie/test: 0.3 secundă/test

KB07 Numărul de biţi (100 puncte)

Din **NRBIT.IN** se citesc două numere naturale **N**≤ 12345678şi **P**≤12345678 Calculaţi numărul minim de biţi pe care se poate reprezernta numărul $\mathbf{N^P}$. Datele de ieşire, numărul de cifre, se scrie în **NRBIT.OUT**

Timp de execuţie/test: 0.3 secundă/test

Exemplu:

NRBIT.IN	**NRBIT.OUT**
2 1	2

KB08 Numărul minim de operaţii - I (70 puncte)

Se dau două numere naturale **N**≤ 100000000 şi **P**≤ 100000000. Determinaţi numărul minim de operaţii de înmulţire şi ridicare la pătrat prin care se calculează $\mathbf{N^P}$. Iniţializarea se consideră operaţie. Datele de intrare, numărul N şi P se citesc din **OPMIN.IN**, iar datele de ieşire, numărul minim de operaţii se scrie în **OPMIN.OUT**.
Exemplu:

OPMIN.IN	**OPMIN.OUT**
2 2	3

Timp de execuţie/test: 0.3 secundă/test

KB09 Numărul minim de operaţii - II (100 puncte)

Se dau două numere naturale **N**≤ 100000000, **P**≤ 1000000 şi **K**≤ 100000000. Determinaţi numărul minim de operaţii re ridicare la pătrat şi înmulţire prin care se calculează şi se afişează $\mathbf{N^P}$ **mod K**. *Iniţializarea şi afişărea se consideră operaţii*. Datele de intrare, numărul N şi P se citesc din **OPMIN2.IN**, iar datele de ieşire, numărul minim de operaţii şi valoarea expresiei, se scriu în **OPMIN2.OUT**.

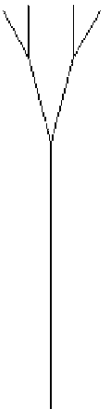
Exemplu: pentru $\mathbf{7^{9007}}$ **mod 561** avem

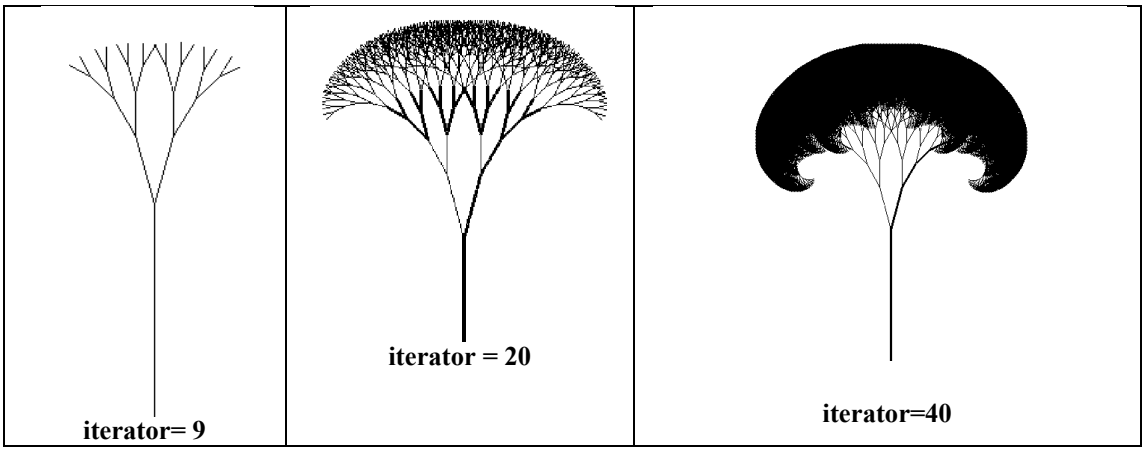
OPMIN2.IN	**OPMIN2.OUT**
7 9007 561	44 226

Timp de execuţie/test: 0.3 secundă/test

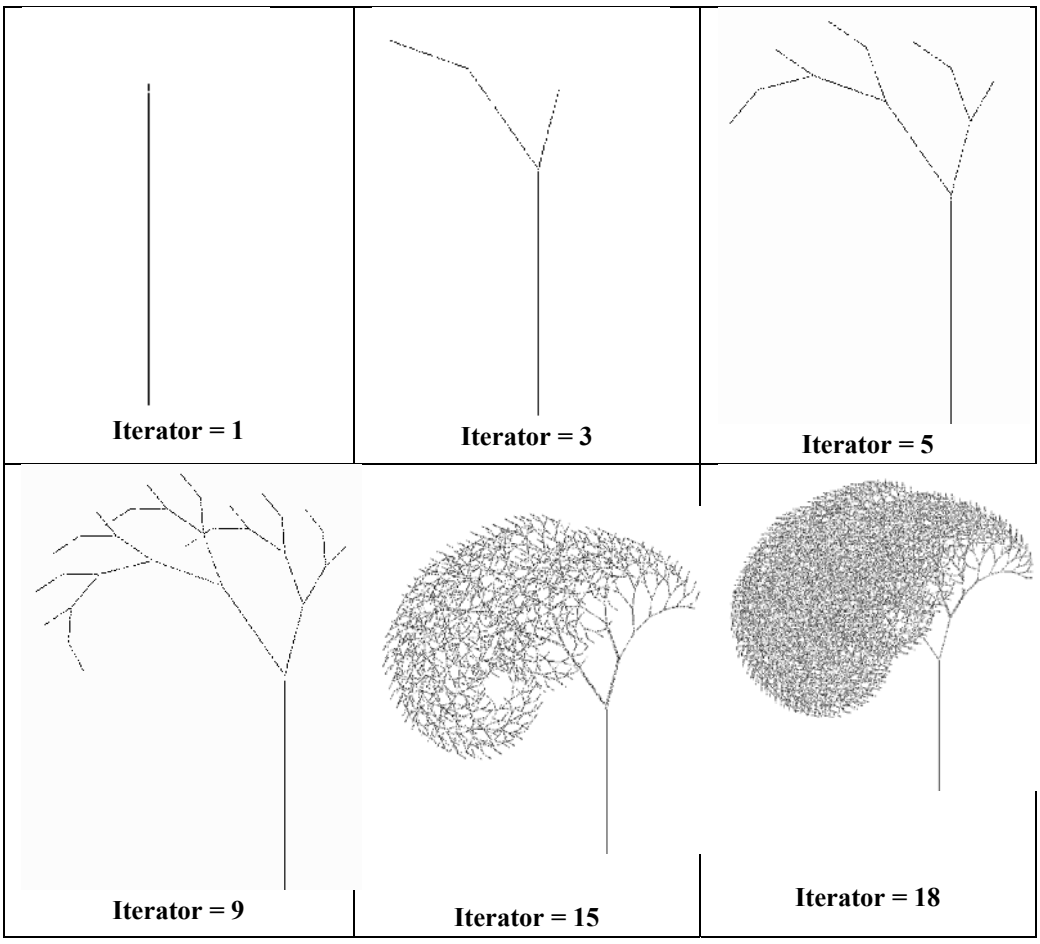
KB10 Grafică recursivă I (80 puncte)

Folosind facilităţile grafice ale mediului **Borland C 3.1** realizaţi următoarea figură

 **iterator = 3**	 **iterator=5**	 **iterator = 7**



KB11 Grafică recursivă II. '*hommage à Cs.K.T.*' **(100 puncte)**



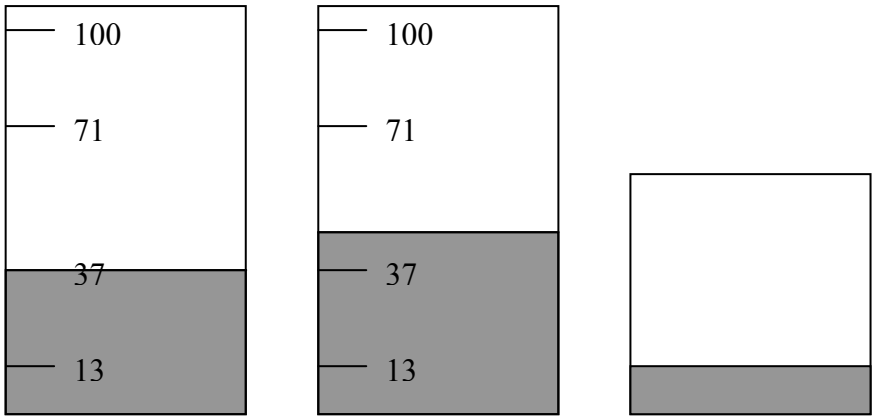
LC01 Măsurători cu sticle (60 puncte)

Se consideră `3` vase. Capacitatea fiecărui vas este de `100` unități (cm^3). Primele două vase au gradații, aceleași pe ambele vase, fiecare gradație indicând

volumul măsurat de la fundul vasului până la acel marcaj. Volumul este trecut lângă marcaj (vezi figura). Inițial primul vas conține 100 de unități de volum de lichid și celelalte sunt goale.

Cerință:

Scrieți un program care să decidă dacă se poate separa o unitate de volum în al treilea vas și dacă e posibil, calculați numărul minim de pași necesari. La fiecare pas, după efectuarea unei operații, cel puțin un vas folosit trebuie să conțină lichid până la o anumită gradație sau să fie gol. În timpul acțiunii putem reține conținutul curent al fiecărui vas.



Date de intrare:

Prima linie a fișierului text `masura.in` conține numărul *n* de gradații ale primelor două vase (*1<=n<=20*). Următoarea linie conține *n* numere întregi în ordine crescătoare, și anume volumele corespunzătoare marcajelor de pe vase. Pentru fiecare marcaj *V* avem *1<=V<=100* și cea mai mare valoare este 100.

Date de ieșire:

Tipăriți DA în prima linie a fișierului `masura.out`, dacă se poate separa o unitate de volum în cel de-al treilea vas și NU în caz contrar. Dacă răspunsul este DA atunci tipăriți numărul minim de pași în linia a doua a fișierului.

Exemplu:

MASURA.IN	**MASURA.OUT**
4 13 37 71 100	DA 8

GCC01. NUMERE ZECIMALE CU PERIOADĂ (100 puncte)

Din fișierul **ZECIMAL.IN** se citesc n perechi de numere (x,y). Să se determine pentru fiecare pereche rezultatul împărțirii lui x la y, specificând și perioada. Datele se vor afișa în fișierul **ZECIMAL.OUT.** Restricții: $0 \leq x \leq 2^{30}$, $0 \leq y \leq 2^{30}$. $1 \leq n \leq 1000$.

ZECIMAL.IN	ZECIMAL.OUT
4 1 1 1 2 1 0 1 6	1 0,5 NU 0,1(6)

GCC02. POLINOM (50 puncte)

Să se afișeze coeficienții polinomului $(x+a_1)(x+a_2)(x+a_3)\ldots(x+a_n)$, unde n<100 este un număr natural nenul, iar a_i sunt numere naturale mai mici decât 100. Din fișierul **POLINOM.IN** se citește de pe prima linie valoarea n, urmată de pe linia 2 de toate cele n valori a_1, a_2, … , a_n. Coeficienții se vor afișa în fișierul **POLINOM.OUT** începând cu coeficientul termenului x^n, apoi coeficientul termenului x^{n-1}, …, coeficientul lui x^0. Fiecare coeficient se va afișa pe linie nouă.

POLINOM.IN	POLINOM.OUT	EXPLICAȚII
4 1 2 3 4	1 10 35 50 24	(X+1)(X+2)(X+3)(X+4) =(X^2+3X^1+2)(X+3)(X+4) =(X^3+6X^2+11X^1+6)(X+4) =(**1**X^4+**10**X^3+**35**X^2+**50**X^1+**24**) Se afișează 1, 10, 35, 50, 24

GCC03. PUTERE (60 puncte)

Să se scrie matematic, în cât mai puține operații, modalitatea de a obține pornind de la valoarea 1, numărul n^m, (1<n<100, 1<m<100) efectuând doar operațiile: ridicare la puterea 2 și înmulțire cu n. Valorile n și m se citesc din fișierul **PUTERE.IN**, iar expresia matematică cu paranteze se va scrie pe prima linie a fișierului **PUTERE.OUT**, urmată de nr de operații pe linia a doua, iar pe linia a treia valoarea lui n^m.

PUTERE.IN	PUTERE.OUT
2 21	((((((1*2)^2)^2)*2)^2)^2)*2 7 operatii 2097152
10 7	((((1*10)^2)*10)^2)*10 5 operatii 10000000

GCC04. RELAȚII DE ECHIVALENȚĂ (50 puncte)

Se citesc din fișierul **RELATIE.IN** n≤100 perechi de elemente notate cu litere mici din alfabetul englezesc, care se găsesc într-o relație de echivalență R. Perechile sunt separate prin caracterul 'R'. Să se determine clasele de echivalență ale relației R în fișierul **RELAȚIE.OUT**. Clasele se vor afișa în ordine alfabetică a elementelor.

RELAȚIE.IN	**RELAȚIE.OUT**
6 CRB CRD ERF ARF GRA CRH	CLASA 1: A E F G CLASA 2: B C D H

BK01 Prieteni (50 puncte)

Într-o clasă cu un efectiv de **N** elevi s-a efectuat un sondaj sociometric. Fiecare elev a specificat pe o scală de valori din intervalul **(-1000,1000)**, măsura în care îi îndrăgeşte pe fiecare dintre colegii săi. Valorile pozitive desemnează simpatia, iar valorile negative antipatia faţă de colegi. Sondajul are ca scop punerea în evidenţă a grupurilor de prieteni din clasă. Dintr-un grup de prieteni fac parte acei elevi care se simpatizează cel mai mult.

Se cere să se scrie un program care determină grupurile de prieteni din clasă!

Date de intrare

Prima linie a fişierului de intrare ***PRIETENI.IN*** conţine numărul de elevi din clasă: **N (2≤N≤1000).**

Pe fiecare din următoarele **N** linii ale fişierului sunt date câte **N** valori reprezentând gradele de simpatie ale fiecărui elev. Valoarea cu numărul de ordine **j** de pe linia cu numărul de ordine **i** are următoarea semnificaţie: măsura în care elevul al **i**-lea îl simpatizează pe colegul cu numărul de ordine **j**. Gradul de simpatie al unui elev faţă de sine însuşi este întotdeauna **0**. Gradele de simpatie dintr-o linie de fişier trebuie să fie toate distincte!

Date de ieşire

Pe prima linie a fişierului de ieşire ***PRIETENI.OUT*** se va scrie numărul **K** ce reprezintă numărul grupurilor de prieteni formate în urma sondajului. Pe următoarele **K** linii se vor afişa grupurile de prieteni (câte un grup pe o linie), specificând numărul de ordine al fiecărui elev care face parte din grupul respectiv. Valorile de pe o linie vor fi separate prin câte un spaţiu. Membrii unui grup de prieteni pot fi afişaţi în ordine oarecare.

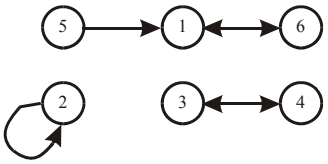
Exemplu

PRIETENI.IN

```
6
0 2 3 4 5 6
-1 0 -3 -4 -5 -6
-1 -2 0 10 1 2   3 4
1 2 10 0 4 5
6 5 4 3 0 1
6 5 4 3 2 0
```

PRIETENI.OUT

```
3
1 5 6
2
```



Problemele au fost propuse de:

- CH – Cristurean Horațiu
- Puni Andrei Paul
- KB – Kovács Barna
- Liţ Claudia
- GCC – Gorea Claudiu Cristian
- BK – Balázs Katalin