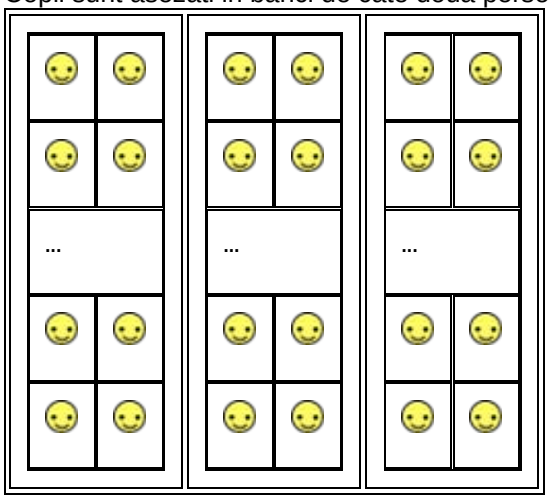
Lista probleme (rezolvati una la alegere)

PB_4 --- Problema de cautare (un mesaj…)

In drum spre scoala un copilas a auzit doi copii mai mari vorbind despre un coleg si prieten bun de-al lui. Cei doi voiau sa-i faca o farsa rautacioasa in prima pauza. Copilasul insa a ajuns dupa ce incepuse ora si a trebuit sa se aseze direct in banca, fara a-si putea avertiza colegul de primejdie. Din fericire i-a venit in minte sa-i trimita din coleg in coleg un bilet cu un mesaj in care sa-l atentioneze despre farsa.

Copii sunt asezati in banci de cate doua persoane. Bancile sunt dispuse in trei coloane astfel:



Un copil poate da fara probleme biletul catre colegul de banca, la fel neobservat de catre profesor poate da biletul catre colegii din fata sau din spatele lui, insa nu si in diagonala, deoarece, intinzandu-se peste banca ar atrage privirea profesorului.

Considerand fragmentul de clasa de mai jos:

a	b
c	d

Elevul *a* poate trimite biletul doar catre *b* sau *c*.

Probleme apar din faptul ca unele locuri in banci pot fi libere, dar si din faptul ca o serie de colegi sunt suparati intre ei si nu-si vorbesc, deci nici nu ar trimite biletul (supararea este reciproca).
De asemenea, trecerea biletelului de pe un rand pe altul este mai anevoioasa, deoarece poate fi vazut foarte usor de catre profesor, de aceea singurele banci intre care se poate face transferul sunt penultimele si ultimele de pe fiecare rand.

Copilul vrea sa scrie pe bilet drumul pe care trebuie sa-l parcurga, de la un coleg la altul, pentru a fi sigur ca nu se rataceste prin clasa si nu mai ajunge la prietenul sau pana la inceputul pauzei.

Se considera prin conventie ca:

- fiecare copil este identificat unic prin numele sau.
- niciun copil nu se numeste 'suparati' sau 'liber'
- locurile libere sunt marcate prin identificatorul 'liber'

Formatul fisierului de intrare este urmatorul. Pe primele linii din fisier se precizeaza asezarea in banci. Fiecare linie cuprinde 6 nume (numele sunt formate doar din litere). Primii doi identificatori corespund primei coloane de banci, urmatorii doi identificatori coloanei din mijloc, si utltimii doi, ultimei coloane de banci.

Dupa ce se termina liniile cu asezare in banci, apare un rand cu identificatorul *suparati*. Sub acest rand, sunt trecuti cate doi elevi (numele lor) care sunt suparati intre ei.

Exemplu de fisier:

```
ionel alina teo eliza carmen monica
george diana bob liber nadia mihai
liber costin anda bogdan dora marin
luiza simona dana cristian tamara dragos
mihnea razvan radu patricia gigel elena
liber andrei oana victor liber dorel
viorel alex ela nicoleta maria gabi
suparati
george ionel
ela nicoleta
victor oana
teo eliza
teo luiza
elena dragos
alina dragos
mesaj: ionel -> dragos
```

Exemplu de drum din fisierul de iesire:

```
ionel > alina v diana v costin v simona v razvan v andrei >> oana ^ radu >
patricia v victor v nicoleta >> maria > gabi ^ dorel ^ elena < gigel ^
tamara > dragos
```

In urma rularii se va afisa drumul parcurs, in ordine cronologica. Daca biletelul merge in cadrul aceluiasi rand, deci catre un coleg de banca, in stanga, se va afisa <, daca merge in dreapta, se va afisa >. Daca biletelul merge spre spatele clasei, intre copiii care transmit biletul se va afisa un *v* (pe post de sagetica in jos), iar daca merge spre fata clasei, se va afisa ^ pe post de sagetica in sus. Cand biletul se deplaseaza spre stanga de pe un rand de banci pe altul, se va afisa <<, iar spre dreapta: >>.

PB_5 --- Problema vaselor cu apa

Consideram ca avem niste vase cu apa colorata. Despre fiecare vas stim capacitatea maxima si cat lichid contine. Pot exista si vase vide. De asemenea pentru combinatia a doua culori de lichide stim ce culoare rezulta din combinatia lor. Pentru combinatiile de culori neprecizate, inseamna ca nu ne intereseaza rezultatul si desi le putem amesteca (uneori e nevoie sa depozitam apa intr-un vas, ca sa facem loc pentru alte mutari) culoarea rezultata nu va aparea in starea solutie niciodata (puteti considera un identificator special pentru acea culoare, de exemplu "nedefinit"). Evident, apa cu culoare nedefinita nu poate fi folosita pentru a obtine alte culori (apa cu culoare nedefinita, amestecata cu orice rezulta in culoare nedefinita).

Lichidul dintr-un vas nu poate fi varsat decat in alt vas (nu dorim sa pierdem din lichid; nu se varsa in exterior).

Formatul fisierului de intrare:

Primele randuri se vor referi la combinatiile de culori. Vor fi cate 3 pe rand, de exemplu:

c1 c2 c3

cu semnificatia ca din combinarea culorii c1 cu c2 rezulta c3 (nu conteaza cantitatea apei amestecate ci doar culoarea ei). Combinatiile sunt simetrice, adica, daca din c1 combinat cu c2 rezulta c3, atunci si din c2 combinat cu c1 rezulta c3.

Sub randurile cu culorile avem un rand cu textul "stare_initiala", dupa care urmeaza starea initiala a vaselor. Pentru fiecare vas se precizeaza cantitatea maxima a acestuia, cata apa are si ce culoare are apa. Toate cantitatile sunt date in litri. In cazul in care cantitatea de apa este 0, lipseste si culoarea. Dupa precizarea starii, initiale, avem textul "stare_finala". Sub acest text pe cate un rand, se specifica o cantitate si o culoare, cu sensul ca in starea finala, pentru fiecare astfel de cantitate (si culoare) precizata trebuie sa existe un vas care sa o contina. Tranzitia consta din turnarea apei dintr-un vas in altul. Se considera ca nu stim sa masuram litrii altfel decat folosind vasele. **Cand turnam lichid putem turna ori pana se termina lichidul din vasul din care turnam, ori pana se umple vasul in care turnam.. Nu se varsa lichid in exterior, lichidul nu da pe afara.** Asfel daca, de exemplu, am un vas cu capacitate 6 si cantitate 3 si unul cu capacitate 4 si cantitate 2, nu putem turna din primul doar un litru, ci doar 2 litri (fiindca sunt 4-2=2 litri liberi in vasul al doilea). In felul asta ramanem cu un litru in primul.
Ne oprim din cautat cand reusim sa ajungem in starea finala: cu alte cuvinte, cand fiecare cantitate de apa colorata specificata in stare se gaseste in cate un vas (nu ne intereseaza cantitatile de apa din restul vaselor)

Exemplu de fisier de intrare:

```
rosu albastru mov
albastru galben verde
mov verde maro
stare_initiala
5 4 rosu
2 2 galben
3 0
7 3 albastru
1 0
4 3 rosu
stare_finala
3 mov
2 verde
```

In fisierul de output se vor afisa starile intermediare, pornind de la starea initiala la starea finala.
Pentru fiecare vase se aloca un id (poate fi numarul de ordine din fisier - de exemplu, vasul cu id-ul 0 este cel cu capacitate 5 si 4 litri de apa rosie)

O stare se va afisa, afisand intai (daca nu e vorba de prima stare) ce miscare s-a facut, in formatul:

```
Din vasul X s-au turnat L litri de apa de culoare C in vasul Y.
```

Apoi se va afisa starea vaselor astfel:

```
id_vas: capacitate_maxima cantitate_apa culoare_apa
```

De exemplu, pentru un succesor al starii initiale de mai sus am avea:

```
Din vasul 0 s-au turnat 1 litri de apa de culoare rosu in vasul 4.
0: 5 3 rosu
1: 2 2 galben
2: 3 0
3: 7 3 albastru
4: 1 1 rosu
5: 4 3 rosu
```

Insa starea initiala s-ar fi afisat in drum fara randul cu mutarea:

```
0: 5 4 rosu
1: 2 2 galben
2: 3 0
3: 7 3 albastru
4: 1 0
5: 4 3 rosu
```

Nu ne preocupam de corectitudine gramaticala ("culoare rosie" in loc de "culoare rosu").

Cand se afiseaza "drumul" de stari, se afiseaza si prima stare in formatul de mai sus (cu id-urile vaselor), dar fara mesajul mutarii.

Un exemplu de distributie a apei intr-o stare finala este (observam ca nu am precizat culoarea pentru cantitate 0):

```
0: 5 5 rosu
1: 2 1 galben
2: 3 1 albastru
3: 7 2 verde
4: 1 0
5: 4 3 mov
```

Indicatie: nu precizati toate starile finale posibile, ci faceti o functie care testeaza daca o stare indeplineste conditia ceruta.

Alte precizari:

- Pentru fisierele de input nu e obligatoriu sa folositi nume reale de culori
- Nu e obligatoriu sa se foloseasca toate culorile in starea finala.

PB_6 --- Micul vrajitor

Un mic vrajitor a pornit la drum sa gaseasca o piatra magica. Numai ca piatra magica se gaseste intr-o pestera la fel de magica. Pestera e de forma dreptunghiulara impartita in parcele patratice, fiecare de o anumita culoare. La intrarea in pestera un batran vrajitor ii ofera o pereche de cizme de culoarea parcelei care se afla la intrarea in pestera, si il povatuieste sa nu calce niciodata cu o pereche de cizme de o culoare pe o parcela de alta culoare fiindca in mod sigur va muri. Micul vrajitor mai poate sa poarte in desaga o pereche in plus de cizme (o pereche de cizme de rezerva). Batranul de asemenea il atentioneaza ca din cauza energiei terenului vrajit, cizmele se strica dupa 3 parcele parcurse, iar daca nu le schimba pana se strica de tot, va ramane descult in urmatoarea parcela si iar va muri. Unele parcele pot contine o pereche de cizme. Micutul vrajitor poate lua acea pereche de cizme si sa o puna in desaga ori sa-si schimbe cizmele curente pentru intrarea intr-o noua parcela de alta culoare (la schimbarea cizmelor, daca are desaga goala le va pune pe cele vechi in desaga, altfel are de ales intre a le arunca pe acestea ori a le arunca pe cele din desaga si a le pune pe acestea in locul lor). Se considera ca schimbarea cizmelor se face dupa parasirea parcelei curente dar imediat inainte de intrarea intr-o noua parcela (deci sa zicem, intermediar, pe granita - dar nu e nevoie sa considerati granita ca o stare aparte; pasul si scimbarea cizmelor vor fi vazute ca o tranzitie unitara). Micul vrajitor atunci cand are desaga goala va lua intotdeauna cizmele din parcela sa le puna in desaga; de asemenea nu va arunca cizmele din desaga daca nu are altele pe care sa le puna in loc (fie cele incaltate, fie cele gasite in parcela). Insa daca cizmele din desaga sunt de aceeasi culoare cu cele din parcela si sunt nepurtate (numarul de purtari e 0) atunci nu le schimba (nu are de ce, ajunge la 2 stari succesoare posibile identice).

Se considera ca daca au fost luate cizmele dintr-o parcela, dupa plecarea vrajitorului, prin magie apare o noua pereche de aceeasi culoare in acelasi loc (deci daca a luat o pereche de acolo, dar dupa niste pasi a ajuns tot in acea parcela ar gasi o pereche de cizme la fel cu cele luate anterior, si le-ar putea folosi). Nu poate lua insa mai mult de o pereche de cizme dintr-o parcela (adica nu poate sa schimbe si cizmele din desaga si pe cele pe care le avea incaltate cu cele gasite in parcela). Cizmele gasite intr-o parcela sunt intotdeauna noi.

Vrajitorasul tinde spre reinnoirea cizmelor. Daca are in desaga cizme de culoarea celor gasite in parcela, sau poarta cizme de culoarea celor gasite in parcela le va reinnoi daca acestea nu sunt noi deja.

Pentru a nu obtine 2 succesori identici pentru anumite stari, luati in vedere faptul ca nu are sens sa-si schimbe cizmele din desaga cu cele incaltate daca sunt de aceeasi culoare si au acelasi numar de purtari.

La intrarea in pestera se considera ca s-a facut un pas, deci cizmele in starea initiala au numarul de purtari egal cu 1.

Vrajitorul se poate misca doar pe linie si coloana, nu si pe diagonala

Atentie scopul nu e doar sa ajunga la piatra ci si sa iasa cu ea din pestera (intoarcerea la nodul initial). Dupa cum se vede micul vrajitor e supus la grea incercare si numai voi il puteti ajuta sa gaseasca drumul catre piatra magica.

Aveti un pic mai jos un exemplu de fisier de intrare. Acesta consta din doua matrici separate printr-o linie goala. Liniile unei matrici sunt fiecare pe cate un rand nou. Elementele pe linie se separa prin spatii (se poate considera ca fiecare element al matricii e de un singur caracter). Prima matrice reprezinta harta efectiva cu culorile parcelelor. A doua matrice este cea care arata ce obiecte se gasesc in parcelele respective. Tot in a doua matrice e marcat locul de pornire cu un caracter * si locul unde se gaseste piatra cu un @. In parcela de pornire si in cea cu piatra nu aveti si cizme. Daca o parcela nu contine nimic, in matricea a doua va avea alocat un 0 pe pozitia corespunzatoare ei.

De exemplu:

```
v r r r
v v a v
v a a a


0 r a *
0 0 0 0
0 0 @ v
```

Pentru acest exemplu de fisier de intrare se considera ca se porneste de la coordonatele 0,3 de pe o parcela de culoare r. Piatra se afla la coordonatele 2,2 pe o parcela de culoare a. Parcela de la coordonatele 0,0 este de culoare v si nu contine nimic. Parcela de la coordonatele 0,2 e de culoare r si contine cizme de culoare a.
Desi programul cunoaste harta, se considera ca vrajitorul nu stie cum arata terenul pana nu il descopera singur, mergand prin pestera.

De exemplu pentru fisierul de intrare:

```
g r v v v v
a r r r r a
a v a r r a
v v a r r a
g g a r r a
v r r g v v
r r a a a a


0 * 0 0 0 0
g a 0 0 0 0
0 r 0 0 0 r
0 0 g 0 0 0
v 0 @ 0 0 0
0 0 0 0 0 g
0 0 0 0 0 0
```

O solutie posibila e cea de mai jos: (acesta ar fi si continutul fisierului de iesire)

```
Pas 0). Incepe drumul cu cizme de culoare r din locatia(0,1). Incaltat: r
(purtari: 1). Desaga: nimic. Fara piatra.


Pas 1).Paseste din (0,1) in (1,1). Incaltat: r (purtari: 2). Desaga: nimic.
Fara piatra.


Pas 2).A gasit cizme a. Schimba cizmele din desaga cu cele din patratel si
porneste la drum. Paseste din (1,1) in (1,2). Incaltat: r (purtari: 3).
Desaga: a (purtari: 0). Fara piatra.
```

Pas 3).I s-au tocit cizmele r. Incalta cizmele din desaga si porneste la drum. Paseste din (1,2) in (2,2). Incaltat: a (purtari: 1). Desaga: nimic. Fara piatra.

Pas 4).Paseste din (2,2) in (3,2). Incaltat: a (purtari: 2). Desaga: nimic. Fara piatra.

Pas 5).A gasit cizme g. Schimba cizmele din desaga cu cele din patratel si porneste la drum. Paseste din (3,2) in (4,2). Incaltat: a (purtari: 3). Desaga: g (purtari: 0). Fara piatra.

Pas 6).I s-au tocit cizmele a. Incalta cizmele din desaga ia piatra si porneste la drum. Paseste din (4,2) in (4,1). Incaltat: g (purtari: 1). Desaga: nimic. Cu piatra.

Pas 7).Paseste din (4,1) in (4,0). Incaltat: g (purtari: 2). Desaga: nimic. Cu piatra.

Pas 8).A gasit cizme v. Muta cizmele incaltate in desaga si le incalta pe cele din patratel si porneste la drum. Paseste din (4,0) in (3,0). Incaltat: v (purtari: 1). Desaga: g (purtari: 2). Cu piatra.

Pas 9).Paseste din (3,0) in (3,1). Incaltat: v (purtari: 2). Desaga: g (purtari: 2). Cu piatra.

Pas 10).Paseste din (3,1) in (2,1). Incaltat: v (purtari: 3). Desaga: g (purtari: 2). Cu piatra.

Pas 11).I s-au tocit cizmele v. A gasit cizme r. Incalta aceste cizme si porneste la drum. Paseste din (2,1) in (1,1). Incaltat: r (purtari: 1). Desaga: g (purtari: 2). Cu piatra.

Pas 12).Paseste din (1,1) in (0,1). Incaltat: r (purtari: 2). Desaga: g (purtari: 2). Cu piatra.

A iesit din pestera.

Indicatie: pentru a face usor afisarea (dar si verificarea solutiei) este bine sa aveti in cadrul starii ce defineste un nod si un camp care sa reprezinte cazul prin care a ajuns de la o stare la alta (de exemplu: caz 1 a facut un pas simplu, caz 2 a gasit piatra, caz 3 a incaltat cizmele din desaga etc. pentru a nu fi nevoiti sa comparati starea anterioara cu cea curenta) De exemplu, la mine in program ultimul camp dintr-o structura st reprezinta cazul. Solutia scrisa ca lista este: [(0,1,r,1,0,0,0,0), (1,1,r,2,0,0,0,1), (1,2,r,3,a,0,0,7), (2,2,a,1,0,0,0,12), (3,2,a,2,0,0,0,1), (4,2,a,3,g,0,0,7), (4,1,g,1,0,0,1,9), (4,0,g,2,0,0,1,1), (3,0,v,1,g,2,1,6), (3,1,v,2,g,2,1,1), (2,1,v,3,g,2,1,1), (1,1,r,1,g,2,1,11), (0,1,r,2,g,2,1,1)]
Atentie nu e obligatoriu sa memorati starile neaparat in acest fel; este doar o indicatie; orice mod corect de reprezentare alternativa este acceptat.

PB_7 --- Evadarea lui Mormolocel

O broscuta mica de tot statea pe o frunza la fel de mica, ce plutea alene pe suprafata unui lac. Broscuta, spre deosebire de alte surate de-ale sale nu stia sa inoate si nu-i placea apa si poate de aceea isi dorea tare mult sa scape din lac si sa ajunga la mal. Singurul mod in care putea sa realizeze acest lucru era sa sara din frunza in frunza.

Forma lacului poate fi aproximata la un cerc. Coordonatele frunzelor sunt raportate la centrul acestui cerc (deci originea axelor de coordonate, adica punctul (0,0) se afla in centrul cercului). Lungimea unei sarituri e maxim valoarea greutatii/3. Din cauza efortului depus, broscuta pierde o unitate de energie(greutate) la fiecare saritura. Se considera ca pierderea in greutate se face in timpul saltului, deci cand ajunge la destinatie are deja cu o unitate mai putin. Daca broscuta ajunge la greutatea 0, atunci moare.

Pe unele frunze exista insecte, pe altele nu. Cand broscuta ajunge pe o frunza mananca o parte din insectele gasite si acest lucru ii da energie pentru noi sarituri. In fisierul de intrare se va specifica numarul de insecte gasite pe fiecare frunza. Daca broscuta mananca o insecta, ea creste in greutate cu o unitate. Atentie, odata ce a mancat o parte din insectele de pe o frunza, aceasta ramane bineinteles fara acel numar de insecte. O tranzitie e considerata a fi un salt plus consumarea insectelor de pe frunza pe care a ajuns.

Formatul fisierului este:

```
raza
GreutateInitialaBroscuta
id_frunza_start
identificator_frunza_1 x1 y1 nr_insecte_1 greutate_max_1
...
identificator_frunza_n xn yn nr_insecte_n greutate_max_n
```

De exemplu, pentru fisierul de intrare:

```
7
5
id1
id1 0 0 0 5
id2 -1 1 3 8
id3 0 2 0 7
id4 2 2 3 10
id5 3 0 1 5
id6 -3 1 1 6
id7 -4 1 3 7
id8 -4 0 1 7
id9 -5 0 2 8
id10 -3 -3 4 10
```

Un exemplu de drum este:

1)Broscuta se afla pe frunza initiala id1(0,0). Greutate broscuta: 5

2)Broscuta sarit de la id1(0,0) la id2(-1,1). Broscuta a mancat 2 insecte. Greutate broscuta: 6

3)Broscuta sarit de la id2(-1,1) la id6(-3,1). Broscuta a mancat 1 insecte. Greutate broscuta: 6

4)Broscuta sarit de la id6(-3,1) la id8(-4,0). Broscuta a mancat 1 insecte. Greutate broscuta: 6

5)Broscuta sarit de la id8(-4,0) la id9(-5,0). Broscuta a mancat 1 insecte. Greutate broscuta: 6

6)Broscuta a ajuns la mal in 5 sarituri.

Drumurile vor trebui afisate in ordinea costului, unde costul e dat de distanta parcursa.

Outputul de mai sus arata si formatul fisierului de ieisire. Pentru fiecare pas se pune numarul de ordine urmat de o paranteza. Pentru starea initiala si finala avem afisaj diferit asa cum se vede mai sus. Pentru celelalte stari afisam de unde pana unde a sarit broscuta, apoi cat a mancat si care e noua greutate.

PB_8 --- Problema unui lacat

Se considera un lacat cu un numar N de incuietori legate intre ele, puse una in continuarea celeilalte astfel incat cu o cheie speciala se pot accesa toate incuietorile in acelasi timp si se pot incuia/descuia dupa caz. O cheie e impartita pe un numar de zone egal cu numarul de incuietori. O astfel de zona poate incuia descuia sau lasa in aceeasi stare o incuietoare, independent de efectul celorlate portiuni ale cheii asupra celorlalte incuietori. O cheie e codificata printr-o secventa de litere din multimea {d,g,i}, unde d inseamna ca are rolul de a descuia, i are rolul de a incuia iar g e gol, efect nul, lasa incuietoarea corespunzatoare asa cum a gasit-o. O incuietoare poate fi de mai multe ori inchisa, adica, daca a fost incuiata de N ori va trebui sa fie descuiata de N ori ca sa fie deschisa. O incuietoare deja descuiata daca e deschisa de cheie va ramane tot descuiata.

De exemplu daca avem 5 incuietori si primele trei sunt inchise (o data), celelalte 2 deschise si aplicam cheia dgidi

i1	i1	i1	d	d
d	g	i	d	i
d	i1	i2	d	i1

Initial toate incuietorile sunt inchise o data
Sa se gaseasca numarul minim de chei folosite astfel incat sa putem descuia lacatul.

Fisierul de intrare cuprinde codul fiecarei chei in parte sub forma unui termen format din literele d,g,i.

Exemplu de fisier de intrare:

```
idddgii
idiidid
iidddii
iddgiig
dgggiid
didiigg
gigddgg
gdggggg
gggiggd
dgggddg
```

In cadrul afisarii soutiei se va arata la fiecare pas starea incuietorii si cheia folosita.

Exemplu de afisare:

```
Initial: [inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1)]
1) Incuietori:
[inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1)].
Folosim cheia: [g,d,g,g,g,g,g] pentru a ajunge la
[inc(i,1),inc(d,0),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1)].
2) Incuietori:
[inc(i,1),inc(d,0),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(i,1)].
Folosim cheia: [d,g,g,g,i,i,d] pentru a ajunge la
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(i,2),inc(i,2),inc(d,0)].
```

```
3) Incuietori:
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(i,2),inc(i,2),inc(d,0)].

Folosim cheia: [d,g,g,g,d,d,g] pentru a ajunge la
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(d,0)].

4) Incuietori:
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(i,1),inc(i,1),inc(d,0)].

Folosim cheia: [d,g,g,g,d,d,g] pentru a ajunge la
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(d,0),inc(d,0),inc(d,0)].

5) Incuietori:
[inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(d,0),inc(d,0),inc(d,0)].

Folosim cheia: [g,i,g,d,d,g,g] pentru a ajunge la
[inc(d,0),inc(i,1),inc(i,1),inc(d,0),inc(d,0),inc(d,0),inc(d,0)].

6) Incuietori:
[inc(d,0),inc(i,1),inc(i,1),inc(d,0),inc(d,0),inc(d,0),inc(d,0)].

Folosim cheia: [i,d,d,g,i,i,g] pentru a ajunge la
[inc(i,1),inc(d,0),inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(d,0)].

7) Incuietori:
[inc(i,1),inc(d,0),inc(d,0),inc(d,0),inc(i,1),inc(i,1),inc(d,0)].

Folosim cheia: [d,g,g,g,d,d,g] pentru a ajunge la
[inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0)].

Incuietori(stare scop):
[inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0),inc(d,0)]
```

In afisare in fata tuplului se scrie si un inc, adica inc(stare_incuietoare, numar)

Pentru fisierul de mai sus poate dura vreo 4 secunde pana sa va afiseze rezultatul. Incercati initial sa-l faceti sa mearga pe un fisier mai simplu, de exemplu:

```
iid

did

gdg
```

Un rezultat posibil ar fi:

```
Initial: [inc(i,1),inc(i,1),inc(i,1)]

1) Incuietori: [inc(i,1),inc(i,1),inc(i,1)].

Folosim cheia: [g,d,g] pentru a ajunge la [inc(i,1),inc(d,0),inc(i,1)].

2) Incuietori: [inc(i,1),inc(d,0),inc(i,1)].

Folosim cheia: [d,i,d] pentru a ajunge la [inc(d,0),inc(i,1),inc(d,0)].

3) Incuietori: [inc(d,0),inc(i,1),inc(d,0)].

Folosim cheia: [g,d,g] pentru a ajunge la [inc(d,0),inc(d,0),inc(d,0)].

Incuietori(stare scop): [inc(d,0),inc(d,0),inc(d,0)]

S-au realizat 3 operatii.
```

PB_9 --- Multe autobuze

Se considera o multime de statii intr-un oras. Pentru fiecare statie cunoastem autobuzele care opresc acolo. Se considera ca fiecare statie de fapt are 2 opriri cate una pentru fiecare sens al autobuzului (orice autobuz circula in ambele sensuri). Un om vrea sa stie ce autobuze trebuie sa ia si prin ce locatii intermediare trebuie sa treaca pentru a ajunge dintr-o anume statie in alta. Se considera ca doreste sa parcurga cat mai putine statii, de asemenea daca in cadrul drumului sau, prin statiile vecine s1 si s2 circula atat autobuzul a1 cat si autobuzul a2, va prefera sa continue sa circule cu a1 decat sa-l schimbe cu autobuzul a2.
Dupa calcularea solutiei se va afisa si timpul aproximativ petrecut in fiecare statie, respectiv timpul petrecut asteptand. Se considera ca timpul mediu de asteptare a unui autobuz cu numarul pana in 200 e de 5 minute. Autobuzul 200 vine mai rar, in medie se asteapta 10 minute. Iar pentru 300 timpul mediu e de 20 minute. Durata parcurgerii a doua statii e aproximativ 3 minute. La final se va afisa si timpul total al drumului (vedeti exemplele de mai jos).

Fisierul de intrare cuprinde pe primul rand:

```
sursa -> destinatie
```

Apoi, de la randul al doilea incolo, fiecare statie in parte, urmata de numarul de autobuze ce trec pe acolo, urmat de lista autobuzelor respective. Dupa informatiile pentru ultima statie, apare cuvantul 'legaturi' (se garanteaza ca nicio statie nu are acest nume) care denota inceperea celei de-a doua zone a fisierului in care avem enumerate fiecare doua statii vecine (deoarece autobuzele circula in ambele sensuri, legatura e reciproca.

Exemplu de fisier de intrare:

```
'Strada Pisicilor' -> 'Strada Soriceilor'
'Strada Pisicilor' 2 100 105
'Strada Lalelelor' 3 100 103 200
'Strada Caselor' 3 200 104 103
'Bulevardul Copacilor' 4 104 100 103 300
'Parcul Frunza Verde' 2 200 108
'Muzeul Dinozaurilor' 2 200 300
'Strada Trandafirilor' 2 300 200
'Cartierul Zorelelor' 3 300 104 105
'Strada Soriceilor' 1 200
'Strada Florilor' 3 300 104 200
'Turnul Porumbeilor' 2 105 200
'Strada Vrabiilor' 1 105
'Strada Pitigoilor' 5 200 100 104 105 300
'Strada Aricilor' 1 108
'Strada Piticilor' 1 108
legaturi
'Strada Pisicilor' 'Strada Caselor'
'Strada Pisicilor' 'Strada Pitigoilor'
'Strada Pisicilor' 'Strada Lalelelor'
'Strada Caselor' 'Strada Pitigoilor'
'Strada Caselor' 'Strada Lalelelor'
'Strada Caselor' 'Bulevardul Copacilor'
'Strada Lalelelor' 'Bulevardul Copacilor'
'Strada Lalelelor' 'Parcul Frunza Verde'
```

```
'Parcul Frunza Verde' 'Muzeul Dinozaurilor'
'Bulevardul Copacilor' 'Muzeul Dinozaurilor'
'Bulevardul Copacilor' 'Cartierul Zorelelor'
'Muzeul Dinozaurilor' 'Strada Trandafirilor'
'Cartierul Zorelelor' 'Strada Trandafirilor'
'Strada Trandafirilor' 'Strada Soriceilor'
'Turnul Porumbeilor' 'Strada Soriceilor'
'Turnul Porumbeilor' 'Cartierul Zorelelor'
'Turnul Porumbeilor' 'Strada Florilor'
'Strada Pitigoilor' 'Strada Florilor'
'Strada Florilor' 'Cartierul Zorelelor'
'Strada Pitigoilor' 'Strada Vrabiilor'
'Turnul Porumbeilor' 'Strada Vrabiilor'
'Strada Pitigoilor' 'Bulevardul Copacilor'
'Strada Aricilor' 'Strada Piticilor'
'Strada Aricilor' 'Parcul Frunza Verde'
```

In cadrul afisarii solutiei se va arata la fiecare pas statia de unde a venit, cat a asteptat acolo, statia in care a ajuns si cu ce autobuz, asa cum se vede in exemplul de mai jos.

Exemplu de afisare:

```
Initial: Strada Pisicilor
1) Din : Strada Pisicilor (asteapta aprox. 5min). ajunge in Strada Pitigoilor in 3min cu autobuzul 105.
2) Din : Strada Pitigoilor (asteapta aprox. 10min). ajunge in Strada Florilor in 3min cu autobuzul 200.
3) Din : Strada Florilor (asteapta aprox. 0min). ajunge in Turnul Porumbeilor in 3min cu autobuzul 200.
4) Din : Turnul Porumbeilor (asteapta aprox. 0min). ajunge in Strada Soriceilor in 3min cu autobuzul 200.
A ajuns la destinatia Strada Soriceilor in aproximativ 27min.
```

Alt exemplu de afisare:

```
Initial: Strada Piticilor
1) Din : Strada Piticilor (asteapta aprox. 5min). ajunge in Strada Aricilor in 3min cu autobuzul 108.
2) Din : Strada Aricilor (asteapta aprox. 0min). ajunge in Parcul Frunza Verde in 3min cu autobuzul 108.
3) Din : Parcul Frunza Verde (asteapta aprox. 10min). ajunge in Muzeul Dinozaurilor in 3min cu autobuzul 200.
4) Din : Muzeul Dinozaurilor (asteapta aprox. 0min). ajunge in Strada Trandafirilor in 3min cu autobuzul 200.
5) Din : Strada Trandafirilor (asteapta aprox. 0min). ajunge in Strada Soriceilor in 3min cu autobuzul 200.
A ajuns la destinatia Strada Soriceilor in aproximativ 30min.
```

PB_10 --- Placutele colorate

Se considera o grila dreptunghiulara plasata vertical. Aceasta este impartita in locatii patratice (placute) de diverse culori (culorile vor fi notate cu cate o litera; se presupune ca nu avem mai multe culori decat litere; literele a-z sunt suficiente).

La fiecare pas se alege o zona de o anumita culoare care are cel putin 3 patratele in componenta sa. Pentru a forma o astfel de zona patratelele trebuie sa fie vecine pe orizontala si verticala cu alte patratele din acea zona (nu se considera vecine si pe diagonala). Zona respectiva se elimina si toate patratelele de deasupra ei "cad" in locurile ramase libere (nu putem avea loc liber care sa aiba deasupra patratele). In cazul in care avem o coloana libera intre patratele, coloanele din dreapta se muta la stanga, unind astfel zonele separate de coloana libera. Daca e prima coloana libera, toate coloanele se muta la stanga cu o pozitie. Coloanele libere vor fi permise doar in dreapta reprezentarii.

Scopul este sa nu mai avem patratele in grid.

De exemplu, pentru starea:

```
aaaa
abbb
cccc
aaaa
```

avem zonele:

aaaa a...	.... .bbb	.... cccc	.... aaaa

Pentru genera succesori putem alege oricare dintre zone sa fie eliminata, de exemplu, daca eliminam zona cu "c":

```
####
aaaa
abbb
aaaa
```

Acum, toate a-urile fac parte din aceeasi zona, si ar putea fi eliminate intr-un singur pas. Am notat cu # spatiile ramase vide.

Pentru starea de mai jos, daca eliminam zona de c-uri, coloanele ramase se unesc deoarece nu putem avea coloana vida decat in partea dreapta.

```
aaca
abcb
cccc
aacc
```

rezultatul va fi:

```
####
aa##
aba#
aab#
```

Pentru fisierul de intrare

```
aaaa
abbb
cccc
aaaa
```

Un drum din fisierul de iesire ar contine:

```
aaaa
abbb
cccc
aaaa

####
aaaa
abbb
aaaa
Cost mutare: 1.

####
####
####
bbb#
Cost mutare: 1.

####
####
####
####
Cost mutare: 1.

S-au realizat 3 mutari cu costul 3.
```

Pentru fisierul de intrare:

```
aaa
abb
ccc
aaa
```

Fisierul de iesire ar contine:

```
Nu avem solutie
```

Costul unei mutari este dat 1+(N-K)/N, unde N e numarul total de placute de culoare celor elimitate, iar K este numarul de placute eliminate in acea mutare. Cu alte cuvinte, cu cat eliminam mai multe placute cu atat costul e mai mic. De exemplu, daca din starea:

```
aaaa
abbb
cccc
aaaa
```

eliminam a-urile de sus, costul va fi 1+(9-5)/9=1.44444 (alegeti voi pana la ce numar de zecimale aproximati costul).

PB_11 --- Problema decuparii

Se considera un grid (matrice) de dimensiuni N×M (N linii si M coloane). In grid avem litere de la a la z. Exemplu de grid:

```
abbacbq
ddaabbb
aaaffxc
aabccdc
adddaab
```

Fisierul de intrare va contine gridul initial si starea scop, cu o linie vida intre ele:

```
ddaabbb
aaaffxc
aabccdc
adddaab


abb
axc
```

Mutarile posibile sunt:

- taierea unei coloane (cost: 1+(k/(nr coloane taiate)), unde k reprezinta suma, din intreaga zona decupata, a perechilor de vecini care sunt diferiti intre ei; evident fara a lua si simetricele perechilor; de exemplu, daca am gasit perechea {(0,0), (0,1)} nu vom lua si perechea {(0,1), (0,0)}).
- taierea unei linii (cost: (nr coloane inainte de taiere)/(nr linii taiate))

De exemplu, pentru starea:

```
bbx
aba
aab
add
```

decidem ca ar trebui eliminate primele doua coloane. Daca am elimina intai prima coloana, costul ar fi 1+1/1=2, deoarece avem vecini diferiti doar in perechea {(0,0),(1,0)} si am taiat doar o coloana, iar costul celei de-a doua coloane este tot 1+1/1=2, deci in total costul taierii separate ar fi 4. Dar daca am decupa ambele coloane in acelasi timp, costul ar fi 1+3/2=2.5 (perechile de vecini {(0,0),(1,0)}, {(1,0),(1,1)}, {(1,1),(2,1)}, deci in numar de 3, si am taiat doua coloane).

Prin urmare, in acest caz e mai eficient sa taiem impreuna coloanele

Dar pentru cazul de mai jos in care vrem sa taiem tot primele 2 coloane:

```
abx
aba
abb
abd
acd
```

Costul taierii separate a primeia este 1+0/1=1. Costul taierii celei de-a doua este 1+1/1=2, deci costul total al celor doua taieri este 2. Daca taiem ambele coloane, costul va fi 1+6/2=4 deci in acest caz este mai eficient sa taiem separat coloanele.

Pentru starea:

```
ddaabbb
aaaffxc
aabccdc
adddaab
```

Consideram urmatoarea stare finala:

```
abb
axc
```

Un drum posibil este cel de mai jos (nu neaparat cel de cost minim):

```
daabbb
aaffxc
abccdc
addaab

Am eliminat coloanele 1,2,3

dabb
aaxc
abdc
adab

Am eliminat coloana 0

abb
axc
bdc
dab

Am eliminat linile 2,3

abb
axc
```

PB_12 --- Lupi, capre si verze

Problema se bazeaza pe jocul cunoscut cu taranul, lupul, capra si varza
Taranul vrea sa mute animalele si verzele de pe malul de est pe malul de vest. Presupuneam ca taranul nostru si-a dezvoltat afacerea, si de data asta a cumparat L lupi, C capre si V verze. Nu mai are doar o biata barca, ci un vaporas cu 2 compartimente (A si B) in care incap cate K1 si respectiv K2 elemente.

De asemenea, pe malul celalalt taranul are o magazie cu M locuri (adica, in care incap M elemente).

Atat in compartimente cat si in magazie taranul nu pune niciodata elemente de tipuri diferite (de exemplu, vor fi doar capre sau doar lupi, dar niciodata si lupi si capre). Cand taranul nu se afla impreuna cu caprele sau lupii, acestia mananca ce gasesc, mai exact: caprele mananca verze, lupii mananca capre, si, **numai** cand nu au capre, alti lupi. Daca nu au ce manca, toti stau cuminti. Verzele sunt de asemenea cuminti, nu mananca pe nimeni. In magazie sau in compartimentele din barca nimeni nu va manca pe nimeni fiind mereu elemente de acelasi tip.
Barca nu pleaca fara taran, dar poate pleca cu compartimetele goale. Animalele mananca doar dupa ce a plecat barca

Starea finala e atinsa cand nu mai sunt animale pe malul initial si in plus pe malul opus sunt *minim* animalele cerute in fisier (de exemplu, daca se cer "3 verze 10 capre 1 lupi" este valida o solutie cu "10 verze 10 capre 1 lupi".

In fisierul de intrare se vor mentiona:

- pe prima linie, numerele initiale de verze, capre si lupi
- pe a doua linie (numarul de locuri din compartimente A cu K1 locuri si B cu K2 locuri, numarul M de locuri din magazie)
- pe a treia linie L: N1,N2 C:N3 (cat mananca un lup N1-numarul de lupi, N2-numarul de capre, N3 cat mananca o capra). De asemenea presupunem ca intai mananca caprele (din verze) si apoi lupii (din capre/lupi).
- pe a patra linie un sir de forma "Stare finala: V verze, C capre, L lupi" -reprezinta **minimul** de capre, verze si lupi cu care trebuie sa ramana taranul dupa ce a mutat **toate** entitatile de pe malul initial pe malul opus.

De exemplu avem fisierul de intrare:

```
20 verze 5 capre 2 lupi
3 4 5
L: 1,1 C:2
Stare finala: 14 verze 7 capre 1 lupi
```

Starea initiala se traduce astfel. Pe malul est se afla 20 verze, 5 capre, 2 lupi. Compartimentele din barca au A:3 si B:4 locuri. Magazia are 5 locuri. Lupii mananca fiecare cate o capra (si, respectiv, cand nu au capre, cate un lup). Caprele mananca fiecare cate 2 verze. Dupa ce au fost mutate toate entitatile, pe malul vest trebuie sa fie minim 14 verze, 7 capre, 1 lupi.

Observatii:

- se pot face si diverse optimizari, pentru a limita numarul de succesori. De exemplu: nu are sens sa punem in magazie elemente de un tip, daca pe mal nu se gasesc elemente care ar manca acel tip, sau ar fi mancate de acel tip. Din magazie scoatem elementele numai cand e nevoie (de exemplu, vrem sa manance sau sa fie mancate sau sa protejam alte elemente)
- Nu e urmarita obtinerea unui numar cat mai mare de elemente, ci doar sa indeplineasca minimul cerut. Daca dintr-o stare nu mai pate fi obtinut minimul cerut, nu are sens sa o dezvoltam.
- Lupii mananca lupi numai daca nu au existat capre deloc la acea iteratie (altfel nu se mananca intre ei chiar daca nu sunt suficiente capre pentru toti).
- Ca sa limitam numarul de succesori (care oricum nu duc la o solutie), nu mai generam succesori pentru starile care au mai putine elemente decat minimul cerut in starea finala.
- Animalele mananca doar dupa o plecare a barcii. La ultima stare nu se mai specifica pe malul opus daca animalele au mancat (fiind toate 0)
- Cand avem 0 entitati in compartimente sau magazie, scriem 0 fara tipul lor, dar pe mal vom preciza daca avem 0 elemente de un anumit tip.

Un exemplu de drum:

```
Pe malul de est se gasesc:
Taranul 21 verze 10 capre 2 lupi
Pe malul de vest se gasesc:
0 verze 0 capre 0 lupi, magazie: 0
Barca pleaca de la est la vest cu container A: 3 capre, container B: 4 capre
------------------
Pe malul de est se gasesc:
21 verze 3 capre 2 lupi
Pe malul de vest se gasesc:
Taranul 0 verze 7 capre 0 lupi, magazie: 0
Dupa ce au mancat pe malul est: 15 verze 1 capre 2 lupi
Barca pleaca de la vest la est cu container A: 0, container B: 0
------------------
Pe malul de est se gasesc:
Taranul 15 verze 1 capre 2 lupi
Pe malul de vest se gasesc:
0 verze 7 capre 0 lupi, magazie: 0
Dupa ce au mancat pe malul vest: 0 verze 7 capre 0 lupi, magazie: 0
Barca pleaca de la est la vest cu container A: 3 verze, container B: 4 verze
------------------
Pe malul de est se gasesc:
8 verze 1 capre 2 lupi
Pe malul de vest se gasesc:
Taranul 7 verze 2 capre 0 lupi, magazie: 5 capre
Dupa ce au mancat pe malul est: 7 verze 0 capre 2 lupi
Barca pleaca de la vest la est cu container A: 2 capre, container B: 0
------------------
Pe malul de est se gasesc:
Taranul 7 verze 2 capre 2 lupi
Pe malul de vest se gasesc:
7 verze 0 capre 0 lupi, magazie: 5 capre
Dupa ce au mancat pe malul vest: 7 verze 0 capre 0 lupi, magazie: 5 capre
Barca pleaca de la est la vest cu container A: 2 capre, container B: 4 verze
------------------
Pe malul de est se gasesc:
3 verze 0 capre 2 lupi
Pe malul de vest se gasesc:
Taranul 11 verze 2 capre 0 lupi, magazie: 5 capre
Dupa ce au mancat pe malul est: 2 verze 0 capre 1 lupi
Barca pleaca de la vest la est cu container A: 2 capre, container B: 0
------------------
Pe malul de est se gasesc:
Taranul 3 verze 2 capre 1 lupi
Pe malul de vest se gasesc:
11 verze 0 capre 0 lupi, magazie: 5 capre
Dupa ce au mancat pe malul vest: 11 verze 0 capre 0 lupi, magazie: 5 capre
Barca pleaca de la est la vest cu container A: 3 verze, container B: 1 lupi
------------------
Pe malul de est se gasesc:
0 verze 2 capre 0 lupi
Pe malul de vest se gasesc:
Taranul 14 verze 0 capre 1 lupi, magazie: 5 capre
Dupa ce au mancat pe malul est: 0 verze 2 capre 0 lupi
Barca pleaca de la vest la est cu container A: 0, container B: 0
------------------
Pe malul de est se gasesc:
Taranul 0 verze 2 capre 0 lupi
Pe malul de vest se gasesc:
14 verze 0 capre 1 lupi, magazie: 5 capre
Dupa ce au mancat pe malul vest: 14 verze 0 capre 1 lupi, magazie: 5 capre
Barca pleaca de la est la vest cu container A: 2 capre, container B: 0
------------------
Pe malul de est se gasesc:
0 verze 0 capre 0 lupi
Pe malul de vest se gasesc:
Taranul 14 verze 2 capre 1 lupi, magazie: 5 capre
```