Problema Sort

Fișier de intrare: `sort.in`
Fișier de ieșire: `sort.out`

Fie A un șir de N numere naturale. Construiți un șir B de N numere naturale care respectă următoarele proprietăți:

- Pentru toți $1 \leq i \leq N-1$, $B[i] \leq B[i+1]$.
- Pentru toți $1 \leq i \leq N$, $B[i]$ poate fi obținut din $A[i]$ printr-un număr de permutări circulare ale cifrelor lui $A[i]$. Acest număr poate fi 0, în acest caz $B[i] = A[i]$.

Operația de permutare circulară mută toate cifrele, în afară de prima, cu o poziţie spre stânga şi aduce prima cifră pe ultima poziţie. De exemplu, din numărul 12345 putem obține, printre altele, numerele 34512 și 51234 folosind permutări circulare, dar nu putem obține numărul 14325.

Se garantează că pentru toți $1 \leq i \leq N$, $A[i]$ nu conține cifra 0.

Date de intrare

Prima linie va conține numărul N. A doua linie va conține șirul A, constând din N numere naturale separate prin câte un spațiu.

Date de ieșire

Dacă nu există niciun șir B care respectă proprietățile din enunț, afișați cuvântul NU pe prima linie. Altfel, afișați cuvântul DA pe prima linie. Pe a doua linie afișați orice șir B care respectă proprietățile din enunț.

Subtask-uri

#	Punctaj	Restricții
1	21	$N = 2$
2	22	$1 \leq N \leq 6$
3	57	$1 \leq N \leq 10\,000$

Pentru toate subtask-urile, $1 \leq A[i] \leq 10^8$.

Trebuie să rezolvați corect **toate** testele din cadrul unui subtask pentru a primi punctajul aferent acestuia.

Exemple

sort.in	sort.out
3 2435 2134 1135	DA 2435 3421 3511
3 511 765 4	NU

Explicații

În al doilea exemplu, $B[2]$ poate fi 765, 657 sau 576, dar niciuna din aceste variante nu este mai mică sau egală cu 4.

Problema Kscale

Intrare: `kscale.in`
Ieșire: `kscale.out`

Definim funcția *scale(A, k_1, k_2)* care primește ca argumente o matrice binară A, două numere naturale nenule k_1 și k_2 și întoarce o matrice binară în care fiecare celulă din A a fost înlocuită cu o submatrice de dimensiune $k_1 \times k_2$ de aceeași valoare cu cea originală.

De exemplu, dacă:

$$A = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

$$\text{scale(A, 2, 3)} = \begin{matrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{matrix}$$

Fie B o matrice de dimensiune $N \times M$. Voi trebuie să construiți o matrice A de arie **minimă**, pentru care există k_1 și k_2 astfel încât $scale(A, k_1, k_2) = B$. Orice matrice validă A de arie minimă va fi acceptată.

Date de intrare

Fișierul de intrare `kscale.in` conține pe prima linie două numere naturale n si m reprezentând numărul de linii, respectiv coloane ale matricei B. Următoarele n linii conțin câte m caractere de tip 0 sau 1.

Date de ieșire

Fişierul de ieşire `kscale.out` va conține pe prima linie două numere naturale p și t reprezentând numărul de linii, respectiv de coloane ale matricei A. Pe următoarele p linii se va afișa matricea A în format similar cu cel din fișierul de intrare.

#	Punctaj	Restricții
1	21 puncte	$N = 1$; $1 \leq M \leq 100$
2	18 puncte	$1 \leq N, M \leq 100$ și se garantează că există soluție cu $k_1 = k_2$.
3	49 puncte	$1 \leq N, M \leq 100$
3	12 puncte	$1 \leq N, M \leq 1\,000$

Anumite teste din interiorul unui subtask pot fi grupate, dar nu neaparat toate.

Exemple

kscale.in	kscale.out
1 10 0000001100	1 5 00010
4 6 111000 111000 000111 000111	2 2 10 01

Problema Secretariat

Fișier de intrare: `secretariat.in`
Fișier de ieșire: `secretariat.out`

La Secretariat există N ghișee, numerotate de la 1 la N. Un număr de persoane a format deja N cozi, câte una la fiecare ghișeu.

Fiecare persoană are o singură problemă de rezolvat.Există K tipuri de probleme, numerotate de la 1 la K, unde $K \leq N$. Există cel puțin câte o persoană cu fiecare problemă dintre cele K.

Personalul Secretariatului ar aprecia dacă fiecare coadă ar conține persoane cu un singur tip de problemă. Pentru a realiza acest lucru, poate fi nevoie să rearanjăm cozile. Pentru a rerarranja cozile, se poate face un singur tip de operație:

Operația (i, j) ia prima persoană din coada de la ghișeul i și o plasează în spatele cozii de la ghișeul j. Este permis (deși trist) ca $i = j$.

Dorim să efectuăm un număr minim de operații pentru a satisface proprietatea cerută. Din fericire, asocierea ghișeu-problemă nu este fixată în acest moment și poate fi decisă de noi pentru a minimiza numărul de operații.

Date de intrare

Prima linie va conține numerele N și K, separate printr-un spațiu.

Conținutul următoarelor N linii va fi următorul: linia i va începe cu un număr natural N_i reprezentând numărul de persoane din coada de la ghișeul i, urmat de N_i numere între 1 și K, separate prin câte un spațiu, reprezentând tipul problemelor avute de persoanele aflate la coadă, în ordine dinspre fața cozii înspre spatele ei.

Date de ieșire

Prima linie va conține o singură valoare, ANS reprezentând numărul minim de operații cerut de problemă.

Următoarele ANS linii vor fi de forma i j (unde i și j sunt separate de un spațiu) reprezentând faptul că prima persoană din coada de la ghișeul i merge și se așează în spatele cozii de la ghișeul j.

Subtask-uri

#	Punctaj	Restricții
1	32	$N = K = 2$
2	25	$1 \leq N \leq 1\,000$ și $K = N$
3	43	$1 \leq K \leq N \leq 1\,000$

Pentru toate testele, pentru toți $1 \leq i \leq N$, se respectă $1 \leq N_i \leq 2\,000$, iar totalul de oameni aflați la coadă este cel mult 500 000.

Trebuie să rezolvați corect **toate** testele din cadrul unui subtask pentru a primi punctajul aferent acestuia.

Exemple

secretariat.in	secretariat.out
4 3 4 2 2 1 1 3 3 3 3 4 1 2 1 2 1 1	5 3 4 3 3 1 3 3 1 1 3

Exemplul descrie următoarea situație:

Ghișeul 1**:** 2 2 1 1
Ghișeul 2**:** 3 3 3
Ghișeul 3**:** 1 2 1 2
Ghișeul 4**:** 1

Se poate arăta că nu există soluție cu mai puțin de 5 operații. După cele 5 operații descrise în fișierul de ieșire, cozile vor arăta astfel:

Ghișeul 1**:** 1 1 1
Ghișeul 2**:** 3 3 3
Ghișeul 3**:** 2 2 2 2
Ghișeul 4**:** 1 1

Problema Grile

Intrare: `standard input`
Ieșire: `standard output`

Să facem un exercițiu de imaginație. Să zicem că ai participat la un examen online de tip grilă cu un total de $N = 20$ întrebări, structurat astfel:

- Primele 10 întrebări ușoare, fiecare valorând 2 puncte și având 5 variante de răspuns.
- Următoarele 6 întrebări au fost medii, fiecare valorând 3 puncte și având 5 variante de răspuns.
- Următoarele 4 întrebări au fost dificile, fiecare valorând 5 puncte și având 10 variante de răspuns.

Fiecare întrebare are exact un răspuns corect și poți alege să nu răspunzi deloc la o întrebare.

După ce ai finalizat testul și ai trimis răspunsurile, ai observat că ai primit imediat scorul tău total, deși concursul nu se terminase încă, iar unii concurenți încă rezolvau subiectul. Asta îți dă de gândit: oare cineva care are mai multe conturi poate folosi scorul total pentru a extrage informații despre răspunsurile corecte?

În practică acest lucru nu s-a întâmplat, dar în teorie este perfect posibil. În această problemă trebuie să concepeți o strategie care să minimizeze numărul de conturi pe care ar fi trebuit să le aveți ca să puteți obține punctaj maxim la acest examen imaginar de tip grilă.

Interacțiune

Programul vostru va fi evaluat pe 10 teste, fiecare valorând 10 puncte. În cadrul unui test, programul vostru va simula $T = 200$ de examene. Fiecare examen este structurat conform descrierii de mai sus și are un set de răspunsuri corecte generate aleator uniform.

Inițial veți citi valoarea lui T. Apoi, începeți examenul 1. Cât timp nu ați obținut punctaj maxim pentru examenul curent, puteți trimite un set de răspunsuri către evaluator. Un set de răspunsuri conține exact $N = 20$ numere naturale, codificând răspunsurile voastre pentru fiecare întrebare. Pentru o întrebare cu M variante de răspuns, variantele sunt numerotate de la 0 la $M - 1$ inclusiv. Valoarea -1 semnifică faptul că nu doriți să răspundeți la întrebarea respectivă.

După ce afișați un set de răspunsuri, puteți citi scorul obținut de acest set. Dacă acest scor este maxim, examenul curent s-a terminat, iar evaluatorul va trece tacit la următorul examen (iar voi trebuie să faceți la fel). Altfel, evaluatorul așteaptă următorul set de răspunsuri. Puteți trimite cel mult 200 de seturi de răspunsuri în cadrul unui examen.

Atenție! Protocolul de interacțiune trebuie respectat cu strictețe. Încălcarea protocolului poate afecta evaluarea soluției în feluri neașteptate. În cazul anumitor tipuri de greșeli (spre exemplu, afișați o variantă invalidă pentru o anumită întrebare sau depășiți limita de 200 de seturi per examen), evaluatorul va afișa valoarea -1 în loc de scorul total și va încheia interacțiunea cu 0 puncte. Puteți folosi acest semnal pentru a vă opri programul.

Punctaj

Dacă programul vostru obține punctaj maxim pentru un examen, evaluatorul va reține numărul de seturi pe care le-ați trimis ca să realizați acest lucru. Fie această valoare S. La finalul interacțiunii, programul va face media valorilor S pentru toate cele 200 de examene, fie această valoare S_{med}. Atunci punctajul pentru un test este dat de formula $0.1 \cdot f(S_{med})$, unde

$$f(x) = \begin{cases} 100, \text{ dacă } x \leq 30 \\ 152.5 - 1.75 \cdot x, \text{ dacă } 30 < x \leq 70 \\ 45.62 - 0.22 \cdot x, \text{ dacă } 70 < x \leq 200 \end{cases}$$

Exemple

intrare	ieșire
2 3 4 0 3 4	 0 0 0 0 0 0 0 1 0 1 -1 0 1 0 0 0 1 0 1 0

Explicații

Din cauza limitărilor de spațiu, vom ilustra o interacțiune pentru un test fictiv în care avem $T = 2$ examene cu $N = 4$ întrebări, fiecare cu 2 variante și valorând câte 1 punct. Amintim că toate testele folosite pentru evaluare vor respecta restricțiile descrise în enunț.

Concurentul citește în primul rând de la tastatură valoarea lui T. Apoi trimite un set de răspunsuri pentru primul examen (răspunde cu prima variantă la toate cele 4 întrebări). Evaluatorul îi spune că a obținut astfel scorul 3. Concurentul trimite un nou set de răspunsuri (în care schimbă răspunsul la a patra întrebare) și printr-un noroc semnificativ obține 4 puncte, adică punctaj maxim.

În acest moment, interactorul începe un nou examen, iar concurentul trimite seturi de răspunsuri pentru acesta. Observați că în primul set, concurentul alege să nu răspundă deloc la întrebarea cu numărul 3. După încă două seturi trimise, concurentul obține din nou punctaj maxim, iar interacțiunea se termină.

Concurentul a folosit 2, respectiv 3 seturi pentru a rezolva cele două examene, deci $S_{med} = 2.5$