Problem A. MaratonIME ajuda Pablito

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	64 megabytes

♯ Caloventor tiene miedo... ♭

Benedetto, Nathan

Como toda pessoa culta e letrada sabe, os ratos são os seres mais inteligentes do planeta. Os golfinhos são os segundos.

O MaratonIME sabe da hierarquia entre as espécies, e a utiliza em seu favor. Em geral, quando eles precisam de recurso, eles sabem que é sempre bom ter um inteligente rato auxiliando. Mas os ratos não sentem muita empatia por nós, primatas, e só nos ajudam se eles nos devem favores.

Assim, o MaratonIME decidiu ajudar o ratinho Pablito. Pablito está estudando a genealogia dos ratos, para facilitar clonagem e mapeamento genômico. Felizmente, parte do trabalho já está feita pela forma com a qual os ratos se identificam.

A sociedade dos ratos, historicamente, se organiza de forma matriarcal. No princípio, haviam pequenas famílias, cada uma com sua matriarca na liderança. Naquela época primordial, foi decidido como se organizariam os ratos:

- Cada matriarca recebia como identificação um inteiro maior do que um.
- Cada uma dessas identificações era escolhida de forma que tivesse o mínimo possível de divisores.
- Todos os membros da família tinham a mesma identificação da matriarca.
- A identificação de cada rato nascido a partir daí é o produto da identificação dos pais dele.

Por exemplo, o filho de um rato com identidade 6 com uma rata de identidade 7 é 42.

Pablito precisa, dado dois ratos, saber se eles tem algum ancestral em comum. Sua única ferramenta é o número de identificação de cada ratinho, que é sempre um inteiro positivo, maior do que um, com no máximo 16 dígitos.

Crie um programa que diz se o par de ratos que tem aquela identificação tem algum ancestral em comum.

Input

A entrada começa com um inteiro positivo $t \leq 10^5$.

Depois seguem t linhas, cada uma com dois inteiros a_i e b_i que identificam dois ratinhos.

Toda identificação de um ratinho é um número positivo, maior do que 1, com até 16 dígitos.

Output

Para cada linha, responda "Sim" se os ratinhos a_i e b_i tem um ancestral em comum e "Nao" caso contrário.

Example

standard input	standard output
2 2 4 3 5	Sim Nao

Problem B. Maratonime joga Cîrokime

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Onde já se viu vodka com sabor?

Everaldo, Glauber

Os membros do MaratonIME adoram se divertir, gostam tanto que inventaram um jogo e o nomearam "Cîrokime". O jogo funciona da seguinte forma:

Primeiro n copos com *Cîroc*[1] são enfileirados em uma linha, à frente do i-ésimo copo é escrito um número a_i. É garantido que, para todo $1 \leq i < n$ temos que $a_i < a_{i+1}$. Em seguida os números são cobertos e o jogo começa. O jogador deve então achar o copo que possui um determinado número x, é garantido que tal copo existe. Para isso ele deve escolher um copo i e beber a bebida, em seguida o seu número a_i é revelado e se for igual ao valor x o jogo encerra, caso contrário, o jogador deve escolher outro copo e assim por diante.

"Cîrokime" é uma tradição entre os membros do MaratonIME, eles jogam em toda festa. Na última festa, Sussu ficou em último lugar, teve que beber os n copos e só no fim encontrou o copo certo. Além de ter ficado em último lugar, Sussu passou mal por ter bebido tanto e teve que ser carregado para casa[2]. Mas o DESMAME[3] está marcado para o dia 13 de maio e Sussu quer recuperar sua dignidade, para isso ele quer saber, no pior caso, qual o número máximo de copos que ele terá que beber se jogar de forma ótima.

Input

A primeira linha contém um único inteiro n, o número de copos. A segunda contém n inteiros a_i, os valores escondidos em cada copo.

- $1 \leq n \leq 10^5$
- Para todo i, $1 \leq a_i \leq 10^9$
- Para $i < n$, $a_i < a_{i+1}$

Output

A saída consiste em uma única linha contendo um único inteiro: o número mínimo de copos que Sussu deve beber no pior caso se jogar de forma ótima.

Examples

standard input	standard output
3 2 5 7	2
8 1 2 3 4 5 6 7 8	4

[1] Cîroc é uma marca de vodka feita na França, conhecida por seu alto preço de venda no mercado (mas Balalaika também serve).

[2] História baseada em fatos reais.

[3] Festa pré-BIFE[4].

[4] Jogos universitários dos quais o IME[5] participa.

[5] Instituto de Matemática e Estatística

Problem C. MaratonIME joga Nim

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Ai Fox!

UnionFind, Germano

Você abre seus olhos, mas tudo continua escuro. O mundo está escuro, e tudo balança. Você percebe que está trancado, mas antes de começar a se desesperar, você ouve uma porta abrindo e a luz invade sua visão e te cega por alguns momentos.

Te ajudam a sair, você estava trancado num porta-malas. Você não reconhece as faces mascaradas, mas se lembra que no último treino para Maratona te falaram que "o começo ainda estava por vir". "Então esse é o lendário desafio de iniciação do MaratonIME", você já tinha ouvido boatos desse evento, e se sente honrado de ter sido selecionado.

Depois de entrar em um prédio abandonado, te sentam em uma cadeira velha. O primeiro teste é assistir um jogo de futebol sem nenhuma reação de animação. Fácil. O segundo é instalar Linux em um notebook em menos de 5 minutos. Você já estava preparado, sempre carrega seu pendrive com ArchLinux em caso de emergência. Você enfrenta mais testes, e consegue passar em todos apesar de algumas dificuldades.

Depois de horas, os integrantes tiram suas máscaras, e cada um retira uma moeda do bolso. "Ganhei! E ainda fiquei rico" você pensa, mas percebe que eles colocam as moedas numa mesa na sua frente, divididas em duas pilhas. Renzo, o grande chefe do MaratonIME, pega uma cadeira e senta na sua frente. Vocês devem jogar uma partida de Nim, e se você vencer se torna um membro honorário do MaratonIME, ou seja, ganha um balão.

Nim é um jogo de dois jogadores, que alternam turnos. Duas pilhas de moedas ficam em uma mesa e em uma jogada você pode tirar qualquer quantidade não nula de moedas de uma das pilhas. O último jogador a fazer uma jogada (e deixar as duas pilhas vazias) ganha.

Você começa o jogo. Para não ser injusto foi garantido que existe uma forma de você ganhar. Escreva um programa que vença Renzo 100% das vezes.

Input

Na primeira linha, dois interos, x e y, o tamanho das pilhas, com $0 \leq x, y \leq 10^4$. É garantido que é possível vencer esse jogo.

Interaction Protocol

Na sua jogada, imprima dois inteiros, i e x, onde i é o número da pilha que você vai tirar moedas ($i \in \{1, 2\}$), e x o número de moedas que vai retirar ($x \geq 1$, tal que a pilha i tenha pelo menos x pedras).

Na jogada do Renzo, leia dois inteiros, no mesmo formato que os da sua jogada.

Example

standard input	standard output
2 1 1 1	1 1 2 1

Note

É claro que não fazemos um desafio de iniciação assim :P

No exemplo, temos uma pilha com duas moedas e outra com uma. Você tira uma moeda da primeira pilha, agora qualquer moeda que Renzo retirar, você pode retirar a outra e ganhar.

Lembre-se de, depois de imprimir sua jogada, fazer a descarga da saída, como `fflush(stdout);` em C, `cout.flush();` em C++, ou `sys.stdout.flush()` em Python.

Problem D. MaratonIME joga Xadrez

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Esse problema é chato para baralho.

Kawakami, Marcos

Nosso querido Nathan, quando guri, gostava muito de jogar Xadrez, mas isso já faz muito tempo. Um dia desses ele foi desafiado pelo famigerado @luisgust para uma partida de xadrez e, como gosta muito de um desafio, aceitou. O problema é que Nathan anda se esquecendo de algumas das regras do jogo, então ele pediu sua ajuda para determinar, em alguns momentos do jogo, se uma dada peça inimiga pode ser capturada com um único movimento.

O xadrez, no MaratonIME, é representado como um grid de caracteres. Ao invés de jogar com peças brancas e pretas, os times jogam com peças maiúsculas (representadas por caracteres maiúsculos) e minúsculas (represetadas por caracteres minúsculos). O Nathan escolheu jogar com as peças minúsculas, porque elas jogam antes.

Além disso, usualmente, as posições de um tabuleiro de xadrez são dadas num sistema de coordenadas específico. Toda posição é um par com um caractere entre `a` (inclusive) e `h` (inclusive), representando a coluna, e um inteiro entre `1` (inclusive) e `8` (inclusive), representando a linha. Por exemplo, a posição `d2` se refere à quarta casa (da esquerda para a direita) da segunda linha coluna (de baixo para cima) e a posição `f6` se refere à sexta casa da sexta linha. As peças minúsculas começam do lado de baixo do tabuleiro, ou seja, nas linhas `1` e `2`.

Além disso, o termo posição adjacente de uma dada posição se refere a qualquer uma das posições que compartilham pelo menos um vértice com ela, isto é, duas posições são adjacentes se e somente se o máximo entre a distância entre as linhas e a distância entre as colunas das duas posições for igual a 1. Por exemplo, a posição `c4` é adjacente a 7 posições: `b3`, `b4`, `b5`, `c3`, `c5`, `d3`, `d4` e `d5`.

O MaratonIME, além disso, usa um sistema um pouco simplificado de movimentos de xadrez. Para te ajudar com o seu programa, foi fornecido um manual com os movimentos que cada peça pode realizar para capturar outra peça e as letras que representam cada peça.

- O peão, representado pelos caracteres `p` ou `P`, pode capturar peças adjacentes a ela que compartilhem uma diagonal com ela e estejam à frente dela, ou seja, um peão minúsculo pode capturar uma peça adjacente a ele que tenha uma diagonal em comum com ele e que esteja em uma linha estritamente maior que a dele, por exemplo, um peão na posição `c4` pode capturar peças nas posições `b5` ou `d5`.
- O cavalo, representado por `c` ou `C`, faz movimentos em L em qualquer um dos 8 sentidos possíveis, ou seja, ele anda duas casas em uma direção qualquer e, depois, uma casa em uma direção perpendicular à primeira. Um cavalo na posição `c4` pode capturar, em um movimento, peças nas posições `a3`, `a5`, `b2`, `b6`, `d2`, `d6`, `e3` e `e5`.
- A torre, representada por `t` ou `T`, consegue capturar qualquer peça que tenha alguma coordenada igual a ela e que seja visível por ela em uma linha reta, ou seja, não existe nenhuma peça na linha entre a torre e a peça a ser capturada. Por exemplo, uma torre em `c4` pode capturar qualquer peça na coluna `c` e na linha `4`, desde que não haja nenhuma peça entre elas. Se houver uma torre em `c4` e uma outra peça qualquer em `c6`, a torre não pode capturar ninguém em `c7` nem em `c8`, mas uma torre em `c4` pode capturar uma peça em `a4` desde que não tenha ninguém em `b4`.

- O bispo, representado por `b` ou `B`, consegue capturar qualquer peça que esteja em alguma das diagonais que passam pelo bispo e que seja visivel por ele em uma linha reta, ou seja, uma peça tal que a diferença entre as coordenadas-linha dela e do bispo e as coordenadas-coluna dela e do bispo sejam iguais. Por exemplo, um bispo em `c4` consegue capturar alguém em `f7` desde que não haja ninguém nem em `d5` e em `e6`.
- A rainha, representada por `r` ou `R`, consegue fazer os movimentos da torre e os movimentos do bispo, ou seja, consegue, em um movimento, capturar qualquer peça que tenha alguma coordenada igual à dela ou que esteja em alguma diagonal a qual ela pertence desde que não haja nenhum obstáculo (outra peça) na linha reta entre a rainha e a peça que se quer capturar.
- O rei, representado por `k` ou `K`, consegue capturar qualquer peça que esteja em uma casa adjacente à dele, seguindo a definição de adjacente já explicada.
- O caractere `.` representa uma posição vazia.

Seu programa deve receber uma matriz e a posição da peça inimiga que o Nathan quer capturar e responder `Sim` se é possível capturar essa peça com um movimento e `Nao` caso contrário.

É garatido que existe sempre uma peça inimiga na posição dada, é garantido também que cada time contém, no máximo, dois bispos, duas torres, dois cavalos, 8 peões, um rei e uma rainha.

Input

Cada instância consiste de 8 linhas com 8 caracteres cada. A primeira linha contém os caracteres nas posições `a8`, `b8`, ..., `h8`. A segunda linha contém os caracteres nas posições `a7`, `b7`, ..., `h7`. E assim por diante. Depois da matriz, em uma linha separada, são dadas as coordenadas que indicam a posição da peça inimiga que deve ser capturada.

Output

A saída consiste em uma única linha para cada instância contendo a string `Sim` se é possível capturar a peça ou `Nao` caso contrário.

Examples

standard input	standard output
TCBRKBCT PPPPPPPP .... pppppppp tcbrkbct d8	Nao
....R..... ...p.... .... c5	Sim

Problem E. MaratonIME pega o circular

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Se organizar direitinho, ...

DESCONHECIDO

Para fazer com que a viagem dos alunos ao metrô não seja tão cansativa, a UESP, Universidade do Estado de São Paulo testou uma de suas mais famosas invenções no ônibus da universidade: eles criaram os circulares de Comprimento Interno Infinito! Em tais maravilhas da Engenharia Moderna sempre existam duplas de bancos vazias para que os alunos se sentem e possam conversar um pouco em sua viagem.

Os integrantes do MaratonIME são muito populares, tão populares que eles possuem amigos em todos institutos da UESP, e assim como a grande maioria de estudantes desta universidade eles também têm que pegar o circular após um longo dia aprendendo a consertar Wi-Fi. Por não fazerem esportes como, por exemplo, remo, todos os alunos da UESP se sentam logo após entrarem no circular, formando duplinhas sempre que possível. Pensando nisso, Gi, experiente Maratonista, bola um problema para pensar no caminho ao metrô: Dado um número n que indica quantos institutos estão em uma avenida da UESP e n inteiros a_i que representam a quantidade de pessoas esperando o Circular no ponto do instituto i, a Gi quer saber para cada par l_j, r_j ($l_j \leq r_j$) se caso todas as pessoas que estão esperando em qualquer ponto entre l_j e r_j (inclusive) entrassem num ônibus vazio, seria possível ninguém ficar sozinho num par de bancos, ou seja, todas as pessoas que entraram iriam conseguir ficar sentadas com uma duplinha.

Input

A entrada consiste em uma linha contendo dois inteiros n e m, o número de institutos e o número de perguntas de Gi. Depois disso, seguem-se n inteiros a_i, a quantidade de pessoas esperando nos pontos de ônibus de cada instituto e, após isso, m linhas com dois inteiros cada, l_i e r_i, dizendo os institutos inicial e final contidos na pergunta da Gi.

- $1 \leq n, m \leq 10^5$
- Para todo i, $0 \leq a_i \leq 10^5$
- Para todo i, $1 \leq l_i \leq r_i \leq n$

Output

A saída consiste em uma única linha contendo a string "Sim" se é possível organizar as duplinhas para que ninguém fique sozinho, ou "Nao" caso contrário.

Example

standard input	standard output
5 2 1 4 10 3 2 3 5 2 3	Nao Sim

Note

No primeiro exemplo temos uma avenida da UESP com 5 institutos, nos quais temos a seguinte quantidade de pessoas esperando: 1 4 10 3 e 2. Gi, cheia de dúvidas, se faz 2 perguntas: se é possível formar duplinhas

caso o circular passe em todos institutos entre os institutos de índices 3 e 5, ou ainda se a mesma distribuição é possível para os índices 2 e 3. Para a primeira pergunta temos 15 pessoas no circular, logo não conseguimos dividir todos em duplinhas, alguém deverá sentar-se sozinho no circular, enquanto que no segundo caso temos 14 pessoas, conseguimos então criar 7 duplinhas no ônibus e não deixar ninguém só.

Problem F. MaratonIME vai à aula (ou não)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	64 megabytes

Já passou a lista?

do IME, Aluno

No MaratonIME, assim como em muitos outros grupos, alguns alunos querem apenas comparecer a aulas o suficiente para não reprovarem por faltas (como sabemos, na USP é necessário 70% de presença), porém outros são dedicados e tentam conseguir a maior presença possível, indo para a faculdade mesmo quando estão doentes ou incapacitados. Curiosamente não existem outros tipos de alunos no MaratonIME.

Madeira, um antigo membro do MaratonIME, precisa de ajuda, ele está cursando a matéria MAC4815162342, e compareceu a k das m aulas que já foram ministradas, sendo que MAC4815162342 tem n aulas totais por semestre. Ele te pediu ajuda para descobrir como melhor cumprir seus objetivos, mas como você é novo no MaratonIME, não sabe que tipo de aluno ele é. Com vergonha de perguntar mais, você decide resolver os dois problemas, assim não há como errar.

Input

A única linha de entrada possui três inteiros n, m e k, com $1 \leq n \leq 10^7$ e $0 \leq k \leq m \leq n$.

n é o número de aulas de MAC4815162342 por semestre, m é quantas dessas aulas já foram dadas e k a quantas aulas Madeira compareceu.

Output

Na primeira linha imprima o número mínimo de aulas que Madeira precisa comparecer para conseguir pelo menos 70% de presença, ou -1 se for impossível conseguir 70% de presença.

Na segunda linha imprima a maior porcentagem de presença que Madeira pode conseguir, se for para todas as aulas a partir da próxima. Esse valor deve estar arredondado para baixo para o inteiro mais próximo. Não imprima '%'.

Examples

standard input	standard output
10 5 2	5 70
11 2 1	7 90

Note

No segundo exemplo, a porcentagem máxima que Madeira pode conseguir é 90.9090.

Problem G. MaratonIME vai à raia

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Desacelera Colombooo!!!

do Remo, Gabi

Como todos sabem, os maratonistas são exímios remadores. A raia é um ótimo lugar para se remar, correr e fazer academia. O que poucas pessoas apreciam são as capivaras que moram lá. Capivaras são animais fascinantes! Além de muito bonitas, as capivaras possuem vários comportamentos peculiares. Você sabia que as capivaras podem viver em grupos de até 100 indivíduos?
Numa bela manhã de sol, Yan corria na raia como de costume. Observando as capivaras, percebeu que elas se organizavam numa linha para tomar sol. Cada capivara estava pareada com somente uma outra. Duas capivaras podem estar pareadas se e somente se ambas se enxergam. Uma capivara enxerga todas as outras na direção que olha. Curioso, Yan decidiu representar as capivaras pelas letras A e B, onde A indica que uma capivara está olhando para a direita e B para a esquerda. Por exemplo, a sequência $AABABB$ representa capivaras tomando sol pois é possível parear todas as capivaras seguindo as regras descritas. De tão fascinado, Yan se descuidou e acabou caindo na raia, fazendo uma bagunça com suas representações. Ele conseguiu recuperar algumas, mas agora elas estavam misturadas com outras coisas. Você pode ajudá-lo descobrir se uma dada representação é de capivaras tomando sol?

Input

Cada instância contém uma sequência S de caracteres, composta apenas por 'A' e 'B', uma representação de Yan.

- $1 \leq |S| \leq 10^5$

Output

A saída contém uma única linha. Imprima "Sim" caso a representação seja de capivaras tomando sol ou "Nao" caso contrário.

Example

standard input	standard output
AABABB	Sim

Problem H. MaratonIME vai ao cinema

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	128 megabytes

Melhor que "A Quinta Onda"

do Mundo, Todos

Acredite se quiser, estudos indicam que não é apenas de maratona que vive o MaratonIME.

Num sábado, depois de simular uma prova, nossa turma de heróis resolveu ir aproveitar a tarde numa sessão de cinema. Mas eles não escolheram ir a um cinema qualquer, mas sim ir assistir um filme n-D, ou seja, com n dimensões.

Em certa cena do filme, durante uma perseguição, o personagem principal pulou por cima de uma corrente de estacionamento. É óbvio que no filme este ato foi feito de forma casual e atlética, mas isto deixou Yan, um dos maratonistas, furioso.

"Hollywood fica fazendo essas coisas parecerem fáceis, mas é muito difícil pular por cima de uma corrente!"

O resto da turma, após o filme, argumentou com Yan que o ator com certeza pulou por cima da corrente, e este discordou, afirmando que isso só poderia ser feito por um truque de câmera ou por efeitos especiais.

Para provar seu ponto, ele comprou outro ingresso para o filme e levou instrumentos de medição muito precisos para a sala. O plano de Yan era mostrar que a distância do pé do ator ao chão era menor que a distância da corrente ao chão, o que provaria que o ator de fato não pulou a corrente.

Mas as contas em n dimensões são complicadas. Todo mundo sabe que a distância no plano entre dois pontos (x_0, y_0) e (x_1, y_1) é dada por

$$\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

Muita gente também sabe que a distância entre dois pontos (x_0, y_0, z_0) e (x_1, y_1, z_1) no espaço tridimensional é calculada pela fórmula

$$\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$$

Ambas as fórmulas descrevem a chamada distância euclideana, no caso bi e tri-dimensional. Seu trabalho aqui é, dado três pontos em n dimensões, dizer se o par mais próximo é o entre o pé e o chão, ou entre o chão e a corrente, de acordo com a distância euclideana.

Input

A entrada começa com um inteiro n seguido de três linhas, cada uma representando um ponto num espaço n-dimensional.

Dimensões demais confundem nós, humanos, logo você pode assumir que $n \leq 10^6$

A segunda linha representa as coordenadas do **chão**, e consiste de n inteiros a_i.

A terceira linha representa as coordenadas do **pé do mocinho**, e consiste de n inteiros b_i.

A quarta linha representa as coordenadas da **corrente**, e consiste de n inteiros c_i.

A sala de cinema não era arbitrariamente grande, logo você pode assumir que todas essas medidas tem o módulo menor ou igual a 10^4

Output

Imprima quem está certo nesta história toda, ou seja, imprima "Yan" se a distância do chão pro pé do mocinho é **menor ou igual** a distância do chão pra corrente, ou imprima "MaratonIME" se a corrente estiver mais próxima.

Examples

standard input	standard output
2 0 0 1 1 2 2	Yan
4 0 0 0 0 2 2 2 2 1 1 1 1	MaratonIME

Problem I. MaratonIME vai ao japonês

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	64 megabytes

Matsu?

do MaratonIME, Membro

Nathan e Yan são programadores de todo coração. Eles aplicam seus conhecimentos de algoritmos em contextos que transcendem a maratona, otimizando as coisas mais triviais do dia a dia.

Alguns questionam se eles só não são meio malucos mesmos, e se não era melhor apenas fazer o que precisava ser feito.

De qualquer forma, eventualmente rolam conflitos quando as abordagens deles divergem sobre o que deve ser feito. Isso rola quando eles resolvem ir em restaurantes japoneses.

Todo mundo sabe que o objetivo, num rodízio, é comer a maior quantidade de comidas distintas. Nathan e Yan diferem no como conseguir isso. Ambos os maratonistas, quando sentam para comer uma delícia asiática específica, comem até raspar o prato.

A diferença é que Yan, respeitando a sabedoria dos mestres nipônicos, come na ordem que os pratos chegam, enquanto Nathan alega que os melhores pratos são deixados para o final, para poupar os ingredientes mais caros, e por isso pede que seus pratos venham na ordem invertida.

Dada a ordem `padrão` de chegada dos pratos, e quanto tempo Nathan e Yan vão ficar no restaurante, diga quem vai comer mais alimentos diferentes, ou se, afinal de contas, dá na mesma.

Input

A primeira linha da entrada tem dois inteiros: $0 < n \leq 10^5$, que representa quantos pratos serão servidos, e $0 < T \leq 10^6$, que representa quantos minutos eles vão ficar no restaurante.

Na linha seguinte, seguem n inteiros t_i, com $0 < t_i \leq 10^3$, indicando quanto tempo demora para se comer aquele prato.

Os restaurantes são muito bem geridos, e você pode assumir que sempre que um dos maratonistas termina o prato, o outro já está na mesa.

Output

Se o Yan for comer mais comidas diferentes, imprima "Yan". Se o Nathan for comer mais comidas diferentes, imprima "Nathan". Se ambos forem comer a mesma quantidade de comidas diferentes, imprima "Empate".

Examples

standard input	standard output
10 45 1 2 3 4 5 6 7 8 9 10	Yan
10 45 10 2 3 4 5 6 7 8 9 1	Nathan

Problem J. MaratonIME vai ao japonês (de novo)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Nakato?

do MaratonIME, Membros

Depois de um longo dia com muito treino, os membros do MaratonIME decidiram ir no restaurante japonês. Sim, nós amamos comida japonesa.

Depois de inúmeras barcas de sushi, quando todos já estavam mais que satisfeitos, eles pediram ao sushiman Sussuchi uma última barca. Desafiado, ele respondeu:

– Vocês querem MAIS UMA barca? Vocês terão mais uma barca...

A barca que ele trouxe foi a maior que qualquer maratonista já tinha visto. Alguns dizem que foi a maior barca que já existiu, superando o limite anterior de 10^5 sushis conquistado pela sushiwoman Gioza em 742, em um festival para o monarca da época, Carlos-sama.

Apesar disso, os maratonistas aceitaram o desafio, e conseguiram comer todos os sushis. Depois, estes estavam tão cheios que não aturavam nem tocar uns aos outros. Tendo comido muita comida, os maratonistas estão desnorteados. Ajude-os a descobrir quais pares de amigos estão se tocando, para que possam se afastar.

Os maratonistas são modelados como círculos num plano, e dois maratonistas se tocam se os seus círculos se tocam. É garantido que nenhum par de círculos se intersecta propriamente, ou seja, a área de sua intersecção é nula.

Input

Na primeira linha, um único inteiro n indicando o número de maratonistas ($2 \leq n \leq 1000$).

Cada uma das próximas n linhas tem três inteiros x_i, y_i e r_i, a $(i+1)$-ésima linha descreve o i-ésimo maratonista. (x_i, y_i) são as coordenadas do centro do círculo, e r_i seu raio. ($-10^4 \leq x_i, y_i \leq 10^4$, $1 \leq r_i \leq 2 \cdot 10^4$)

É garantido que a intersecção de qualquer par de círculos tem área nula.

Output

Para cada par de círculos que se tocam, imprima em uma linha os índices desses círculos. As colisões podem ser impressas em qualquer ordem, os índices dos dois círculos podem ser impressos em qualquer ordem também.

Não imprima as colisões mais de uma vez, ou seja, se i se intersecta com j, imprima i j ou j i, mas não ambos.

Example

standard input	standard output
3 0 0 2 5 0 3 10 10 1	1 2

Problem K. MaratonIME vai ao karaokê

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

♯ Me bata, me prenda, faça tudo comigo, ... ♭

e Marrone, Bruno

Depois de milênios repetindo o título do enunciado deste problema, sempre num tom animado e convidativo, Nathan finalmente conseguiu convencer seus colegas a irem ao karaokê. Ele está simplesmente radiante com esta conquista.

Mas há um problema. Depois de tanto ter insistido com seus amigos para irem ao karaokê, Nathan está com medo de passar vergonha na hora de cantar os clássicos da música brasileira:

- Garçon – Reginaldo Rossi
- Boate azul – Joaquim e Manuel
- Coração de papel – Sérgio Reis
- Borbulhas de amor – Fagner
- Você não me ensinou a te esquecer – Fernando Mendes

Para evitar passar vexame, e para não desencorajar seus coleguinhas sobre futuras idas ao karaokê, Nathan resolveu imprimir todas as cifras de músicas que estão disponíveis no karaokê, para poder consultar enquanto canta. No entanto, isso gerou uma quantidade colossal de papel, que ele simplesmente não consegue carregar.

Mas a perseverança e engenhosidade de um programador com dor de cotovelo não é algo a ser subestimado.

Nathan reparou que, afinal de contas, só existem 7 notas musicais. O pessoal que entende do assunto costuma representá-las pelas letras A,B,C,D,E,F e G. Mais ainda, é comum a mesma nota se repetir em sequência. Ele resolveu então, comprimir as músicas, trocando toda ocorrência de notas repetidas pela nota e quantas vezes ela ocorre.

Por exemplo, dado a sequência

$$(A, A, A, B, B, B, C, G, G, G, G, G, G, G, G, G, G)$$

a versão comprimida é

$$A3B3C1G11$$

Infelizmente, Nathan também precisa arrumar seu terno florido e pentear sua barba – dois trabalhos homéricos – e ficou sem tempo para comprimir as notas. Ajude-o a não passar vergonha, escrevendo um programa que faça este serviço.

Input

Cada entrada consiste de apenas uma linha, uma sequência de caractéres S tal que $|S| \leq 10^5$, formada apenas pelas letras A,B,C,D,E,F e G.

Output

Para cada entrada, imprima uma única linha, tal que cada sequência de notas iguais seja substituída pela nota que ocorre e quantas vezes ela ocorre, conforme o exemplo.

Examples

standard input	standard output
ABBGA	A1B2G1A1
AAABBBCGGGGGGGGGG	A3B3C1G11

Problem L. MaratonIME vai ao kart

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Yan, tá vermelho!!!

em Desespero, Passageiros

Certo dia após uma prova, os maratonistas sentiam-se cabisbaixos por conta de resultados ruins. Vendo a situação, Renzo, o coach do MaratonIME, sugeriu que fizessem algo divertido para descontrair a cabeça. Após discussão fervorosa, decidiram correr de kart. Na procura de um kartódromo que fosse viável para todos, acharam o Kartforces, um kartódromo próximo à Cidade Universitária. Entretanto, a pista era muito pequena e comportava apenas dois corredores por bateria. Como bons maratonistas apaixonados por competição, montaram um torneio justo onde todos corriam contra todos, dois a dois, uma única vez. Em cada corrida, o vencedor somava um ponto no placar. Empates são admitidos, isto é, ambos não pontuam. O grande campeão foi o maior pontuador. Sabe-se que N maratonistas estavam presentes e:

- Cada maratonista possui uma habilidade h_i.
- Se $h_i > h_j$ onde $1 \leq i, j \leq N$ e $i \neq j$, então o maratonista i vence a corrida contra o maratonista j.

Você teve acesso a habilidade de todos os maratonistas e agora se pergunta quem foi o grande campeão.

Input

A primeira linha contém um inteiro N, o número de maratonistas. A segunda linha contém N inteiros h_i, a habilidade do $i-$ésimo maratonista.

- $1 \leq N \leq 10^5$
- $0 \leq h_i \leq 10^9$

Output

A saída consiste em um único inteiro i, o maratonista campeão. Caso não seja possível determinar o campeão, imprima -1.

Example

standard input	standard output
3 2 4 6	3

Problem M. MaratonIME volta ao alojamento

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

♯ Renzo, te ligam. Renzo, antendeee ♭

RINGTONES, TOP

Muito frequentado pelos maratonistas mais tradicionais, o acampamento de verão é muito conhecido por suas festas nababescas. Contudo após as festas os competidores sempre devem voltar para seus respectivos alojamentos. Essa volta nem sempre é tranquila, os competidores não muito sóbrios acabam andando de forma estranha e muitas vezes perdendo ou achando dinheiro, além de serem roubados no caminho. O MaratonIME entretanto sempre tem um membro sóbrio em seu grupo, o que garante que eles não vão perder dinheiro no caminho a não ser que sejam roubados. Voltando para o alojamento o MaratonIME sabe que podem ligar para seu coach, Renzo, que irá buscá-los instantaneamente. Entretanto, eles querem a sua ajuda para escrever um programa que, sabendo o caminho, diga qual é o máximo de dinheiro que é possível levar para o alojamento.

Então sua tarefa será escrever um programa que dado um grid N por M e sabendo como os membros do MaratonIME andam quando bêbados diga qual é o máximo de dinheiro que é possível levar para o alojamento. Deve se levar em consideração os seguintes fatos:

1. Os membros do MaratonIME saem da festa e chegam na rua pelo ponto superior esquerdo e começam andando para a direita.
2. Os membros do MaratonIME saem completamente sem dinheiro da festa.
3. Sempre ao chegar ao fim da rua eles descem um ponto e andam na direção contrária. Eles não mudam de direção a não ser nesse caso.
4. Sempre que encontram um ladrão no caminho, este lhes rouba todo o dinheiro acumulado.
5. Eles podem ligar para Renzo a qualquer momento, e Renzo irá buscá-los instantaneamente.

Input

A entrada consiste de dois inteiros N e M, com $1 \leq N, M \leq 10^3$, indicando respectivamente o tamanho da rua e a largura dela, seguidos por N linhas contendo M caracteres que podem ser:

- `'_'` indicando que não há nada naquele ponto da rua.
- `'.'` indicando que há uma moeda de um real naquele ponto da rua.
- `'L'` indicando que há um ladrão naquele ponto da rua.

Output

A saída consiste de um único inteiro indicando o máximo de dinheiro que MaratonIME pode trazer para o alojamento.

Example

standard input	standard output
3 3 __· _·_ L._	2

Note

No exemplo o caminho que os maratonistas fazem é (1, 1) → (1, 2) → (1, 3) → (2, 3) → (2, 2) → (2, 1) → (3, 1) → (3, 2) → (3, 3). Quando estiverem em (2, 2) ou (2, 1), os maratonistas terão 2 moedas, e esse é o máximo que podem conseguir.