Problem A. A Viagem de Papai Noel

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A véspera de natal é um dia muito cansativo para o Papai Noel e os seus ajudantes. Neste dia, o trenó mágico deve ser carregado com os presentes de todas as crianças do planeta. Uma vez carregado, Papai Noel e suas renas partem do Polo Norte, passam por todas as cidades para entregar os presentes em cada residência e então retornam ao Polo Norte.

Para conseguir entregar todos os presentes em um tempo viável, Papai Noel e suas renas devem minimizar a distância percorrida, mas sem deixar de passar por alguma cidade.

Como ajudante do Papai Noel, você foi incumbido de calcular a menor distância possível que Papai Noel e suas renas devem percorrer para completar a árdua tarefa.

Input

A primeira linha da entrada consiste de uma linha contendo um inteiro N ($2 \leq N \leq 20$), indicando o número de cidades na rota de Papai Noel.

A próxima linha contém dois números inteiros x_0 e y_0 ($-1000 \leq x_0, y_0 \leq 1000$), separados por um espaço em branco, indicando as coordenadas do Polo Norte.

As próximas $n-1$ linhas contém, cada uma, um par de números inteiros x_i e y_i ($-1000 \leq x_i, y_i \leq 1000$) separados por um espaço em branco. A i-ésima linha deste conjunto de linhas descreve as coordenadas geográficas da i-ésima cidade.

É garantido que as cidades ocupam coordenadas distintas.

Output

Seu programa deverá fornecer como saída uma única linha contendo a distância mínima necessária para completar o trajeto. Se sua resposta é um valor y e a resposta do juiz é um valor z, sua resposta será considerada correta se $\frac{|y-z|}{\max(1,z)} \leq 10^{-3}$.

Examples

standard input	standard output
3 0 0 1 1 1 0	3.41421
4 0 0 1 0 1 1 0 1	4
5 1 0 3 3 -3 2 -2 -3 3 -2	22.7148

Note

No primeiro exemplo uma possível solução com menor distância é partir da cidade $(0, 0)$ e completar o trajeto através da sequência de cidades: $(1, 0)$, $(1, 1)$ e $(0, 0)$. Somando todas as distâncias, temos: $1 + 1 + \sqrt{2} \approx 3.41421$

No segundo exemplo, uma possível solução percorre as cidades na seguinte ordem: $(0, 0)$, $(1, 0)$, $(1, 1)$,(0,1) e $(0, 0)$, totalizando a distância mínima de 4.

No terceiro exemplo, uma possível solução envolve a trajetória: $(1, 0)$,$(3, 3)$, $(-3, 2)$, $(-2, -3)$, $(3, -2)$ $(1, 0)$ com uma distância mínima de ≈ 22.7148.

Problem B. Buracos de Minhoca

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Um estudante de física está pesquisando os *Buracos de Minhoca*, os quais conectariam pontos distintos do espaço e tempo, por meio da observação de uma região do espaço. Esta região foi dividida em malha quadrada de N^2 pontos, com coordenadas inteiras (i, j), onde $1 \leq i, j \leq N$.

Ele formulou a seguinte hipótese: seria possível se deslocar de um ponto (a, b) para um ponto (c, d) por meio de um buraco de minhoca se, e somente se, $\gcd(a, b) = \gcd(c, d)$, onde $\gcd(x, y)$ é o maior divisor comum entre x e y. Se esta hipótese estiver correta, esta região seria dividida em N setores espacias S_i, dentro dos quais seria possível se mover livremente pelos buracos. O setor S_i seria composto pelos pontos (x, y) tais que $gcd(x, y) = i$.

Auxilie o estudante em sua pesquisa, identificando o número de pontos pertencentes a cada um dos setores espaciais.

Input

A entrada é composta por uma única linha, contendo o valor de N ($1 \leq N \leq 4 \times 10^5$).

Output

Imprima, em uma linha, N inteiros $|S_i|$, separados por um espaço em branco, onde $|S_i|$ é o número de pontos pertencentes ao setor S_i.

Examples

standard input	standard output
2	3 1
3	7 1 1
5	19 3 1 1 1

Note

No primeiro caso, S_1 é composto pelos pontos $(1, 1), (1, 2)$ e $(2, 1)$, enquanto o setor S_2 é formado somente pelo ponto $(2, 2)$.

No segundo caso, $S_2 = \{(2, 2)\}$ e $S_3 = \{(3, 3)\}$. Todos os pontos restantes pertencem ao setor S_1.

Problem C. Cupins vs Tamanduás: A Revanche

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

No *tower defense* "Cupins vs Tamanduás: A Revanche", continuação do clássico "Tamanduás vs Cupins", a premissa básica permanece a mesma: os tamanduás comem os cupins que tentam avançar para uma casa de madeira (um belo banquete!). Cada cupim tem uma massa, em miligramas, e o Tamanduá Bandeira, uma das torres aliadas, consegue consumir M miligramas de cupim antes de ficar cheio e não conseguir comer mais, possibilitando os demais cupins a atravessarem seu caminho.

Há três tipos de cupim no jogo: o Filhote, o Adulto e o Rei, cada um com massa X, Y e Z miligramas, respectivamente. Em uma *wave* (uma sequência ordenada de cupins que marcharão para a casa de madeira), os cupins são consumidos na ordem que chegam aos tamanduás (que também estão organizados em uma fila), de acordo com os seguintes critérios:

1. um tamanduá come um cupim imediatamente, de modo que ele finaliza a "refeição" antes que o próximo cupim da sequência chegue;
2. se o tamanduá puder comer o cupim, ele o comerá; e
3. um tamanduá não pode comer apenas parte de um cupim: se ele não conseguir consumir o cupim inteiro, ele deixará o mesmo passar.

Dadas as massas X, Y e Z dos cupins, a capacidade de consumo M do Tamanduá Bandeira, o número de inimigos N e a ordem S dos cupins na *wave*, determine o número mínimo de tamanduás que devem ser enfileirados para garantir a integridade total da casa de madeira.

Input

A primeira linha da entrada contém os valores dos inteiros X, Y e Z ($1 \leq X, Y, Z \leq 10^5$), separados por um espaço em branco, que correspondem às massas, em miligramas, dos três tipos de cupim do jogo, conforme descrito anteriormente.

A segunda linha da entrada contém os inteiros M e N ($X, Y, Z \leq M \leq 10^5$, $1 \leq N \leq 2 \times 10^5$), separados por um espaço em branco.

A terceira linha contém uma string S de tamanho N, cujos caracteres ('`F`', '`A`' ou '`R`') indicam o tipo de cupim (Filhote, Adulto e Rei, respectivamente) e a ordem que eles chegarão aos tamanduás (o cupim S_i irá logo a frente do cupim S_{i+1}, com $i \in [1, N)$.

Output

Imprima, em um linha, o número mínimo de tamanduás necessários para comer todos os cupins da *wave*.

Examples

standard input	standard output
3 8 10 20 3 AFR	2
1 2 3 10 1 R	1
3 4 5 6 4 RFAF	3
3 8 10 20 8 ARRFRRRF	4

Note

No primeiro caso, o tamanduá 1 come o cupim Adulto e, em seguida, o Filhote, totalizando uma massa consumida de 11 gramas. Deste modo, ele já não consegue ingerir o cupim Rei, cujo massa é de 10 gramas (pois só consegue comer mais 9 gramas), então o deixa para o segundo tamanduá, que o consome, finalizando a *wave*.

No segundo caso há apenas um cupim na *wave*, e um único tamanduá é necessário.

No terceiro caso, o tamanduá 1 consome o cupim Rei e o tamanduá 2 consome o Filhote que o sucede. O tamanduá 3 consumirá o Adulto (pois os tamanduás 1 e 2 podem ingerir apenas 1 ou 3 gramas, respectivamente). O último Filhote não pode ser consumido pelo tamanduá 1, mas o tamanduá 2 pode e o fará, finalizando a *wave*.

Problem D. Durval e os Treinos Intensos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Durval está treinando para correr uma ultra maratona, que são provas com percursos que geralmente superam os 42 quilômetros. O treinamento de Durvalino não tem sido fácil, pois requer a variação de volume e intensidade nas suas corridas, além do forte calor que assola sua cidade durante todo o ano.

Os treinos de Durval ocorrem em uma estrada de terra, que devido aos diversos treinos realizados, possui diversas demarcações de trechos, que começam e terminam em determinados quilômetros dessa estrada. Para tornar seus treinos desafiadores, Durval decidiu escolher alguns desses trechos para que ele corra em alta intensidade, enquanto que os demais quilômetros (entre trechos) serão percorridos em velocidade reduzida, visando sua recuperação cardiorrespiratória.

Como Durval quer que o seu treinamento seja difícil, ele pede sua ajuda. A partir de vários trechos previamente definidos em que Durval pode correr em alta intensidade, sua tarefa consiste em determinar a quantidade máxima de trechos que devem compor o treinamento de Durval. Lembre-se que Durval deve correr pelo menos um quilômetro entre dois trechos intensos para sua recuperação e que os trechos escolhidos não podem compartilhar o mesmo quilômetro da estrada.

Input

A primeira linha da entrada contém um número inteiro N ($1 \leq N \leq 10^5$), indicando a quantidade de trechos de alta intensidade.

Em seguida, existem N linhas, cada uma descrevendo um trecho. A i-ésima linha apresenta dois números inteiros s_i e f_i ($1 \leq s_i < f_i \leq 10^5$) representando os quilômetros de início e fim do i-ésimo trecho, respectivamente.

Você pode considerar que o comprimento total (em quilômetros) da estrada de terra em que Durval treina como o quilômetro final do trecho mais distante do quilômetro 0, de onde ele inicia seu treinamento: $(max(f_i) + 1)$ para $i = 1, ldots, N$.

Output

Imprima um único número inteiro indicando a quantidade máxima de trechos de alta intensidade que Durval pode adotar no seu treinamento.

Examples

standard input	standard output
3 3 5 6 8 10 12	3
4 1 2 1 8 2 8 7 9	2
8 2 4 6 9 7 8 3 5 7 9 9 10 10 11 13 15	4
2 1 9 3 6	1

Note

No primeiro exemplo de teste, Durval adota três trechos para correr em alta intensidade no seu treinamento: $[3, 5]$, $[6, 8]$ e $[10, 12]$. Durval faz sua recuperação nos trechos (de baixa intensidade) definidos pelos quilômetros $[5, 6]$ e $[8, 10]$. Portanto, a resposta é 3.

No segundo exemplo de teste, o treino de Durval contempla os trechos $[1, 2]$ e $[7, 9]$. Portanto, a resposta é 2.

No terceiro exemplo de teste, Durval pode adotar duas configurações distintas de trechos de alta intensidade para seu treinamento:

- $[2, 4]$, $[7, 8]$, $[9, 10]$ e $[13, 15]$;
- $[2, 4]$, $[6, 9]$ e $[10, 11]$ e $[13, 15]$.

Ambas configurações são válidas e maximizam a quantidade de trechos de alta intensidade. Por isso, a resposta é 4.

No quarto exemplo de teste, Durval pode escolher apenas um dos trechos para o seu treino: ou ele escolhe $[1, 9]$ ou $[3, 6]$. Portanto, a resposta é 1.

Problem E. Eds e a Prova Perfeita 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A grande família resolveu atualizar seus conceitos e reformularam a noção de como elaborar a "prova-perfeita": ela precisa abordar todos os assuntos relevantes de forma equilibrada! Os trigêmeos mais velhos, Edsinho, Ed Filho e Edsão puxaram os gêmeos caçulas Edsoff e Edson para elaborarem a prova mais completa possível...

Ajude ao ilustre Costa Júnior a verificar se "a prova perfeita atende duas propriedades: todos os assuntos são abordados e isso ocorre de maneira aproximadamente uniforme."

Considere que a distribuição de assuntos é aproximadamente uniforme se não há mais de 25% de variação entre a quantidade de questões que abordam o assunto menos cobrado e as que abordam o mais cobrado. Claro, todos sabem que os principais assuntos são: *Algebra*, *Aritmetica*, *Backtracking*, *Bigint*, *Combinatoria*, *Estruturas de Dados*, *Geometria*, *Ordenacao*, *Programacao Dinamica*, *SegTrees*, *Teoria dos Numeros* e *Travessia em Grafos*.

Input

A primeira linha consiste de um inteiro N ($1 \leq N \leq 50$), indicando a quantidade de questões da prova.

As N linhas seguintes apresentam, cada uma, os tópicos abordados por cada questão da prova, separados por vírgula. Cada tópico é representado por um string simples com não mais de 20 caracteres.

Output

Para cada tópico que não foi cobrado o suficiente, apresente a mensagem "Falta t!", onde t é o referido tópico. No caso de mais de um tópico, apresente-os um por linha, em ordem de relevância para o objetivo, i.e., na ordem do que foi menos abordado para o mais. Em caso de empate, apresente-os em ordem alfabética. Caso a distribuição seja aproximadamente uniforme, apresente a mensagem "Tudo em Ribas!".

Examples

standard input	standard output
12 Algebra Aritmetica Backtracking Bigint Combinatoria Estruturas de Dados Geometria Ordenacao Programacao Dinamica SegTrees Teoria dos Numeros Travessia em Grafos	Tudo em Ribas!
6 Geometria,Combinatoria Backtracking,Bigint Travessia em Grafos SegTrees,Estruturas de Dados Algebra,Aritmetica Ordenacao,Teoria dos Numeros	Falta Programacao Dinamica!
5 Backtracking,Bigint,Combinatoria,Estruturas de Dados Combinatoria,Estruturas de Dados,Backtracking,Bigint Geometria,Ordenacao,Programacao Dinamica,Algebra Travessia em Grafos,Programacao Dinamica,Aritmetica Travessia em Grafos,Geometria,Ordenacao	Falta SegTrees! Falta Teoria dos Numeros! Falta Algebra! Falta Aritmetica!

Note

No primeiro exemplo, há uma questão para cada tópico.

No segundo exemplo, cada tópico é cobrado uma vez, exceto por "Programacao Dinamica".

No terceiro exemplo, Algebra, Aritmetica, SegTrees e Teoria dos Numeros não ocorrem com o mínimo da frequência desejada.

Problem F. Farejando Arrays Especiais

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Um subarray $A[i..j]$ de um array A é composto por todos os elementos A_k tais que $i \leq k \leq j$. Por exemplo, os subarrays do array $A = [1, 4, 8]$ são [1], [4], [8], [1, 4], [4, 8] e [1, 4, 8] (note que A é um subarray de si mesmo). Um array é dito "especial" caso cada um de seus subarrays contenha pelo menos um elemento que apareça apenas uma vez naquele subarray.

Um exemplo de array especial é $[1, 2, 1]$. Neste caso, todos os subarrays ([1], [2], [1], [1, 2], [2, 1], [1, 2, 1]) contêm pelo menos um elemento que aparece somente uma vez no respectivo subarray.

Em contrapartida, um array **não** especial é $[3, 2, 4, 8, 8, 4, 2, 5]$. Neste caso, o subarray $[2, 4, 8, 8, 4, 2]$ não atende à condição.

Dado um array A, de N inteiros, diga se ele é especial ou não.

Input

A entrada é composta por 2 linhas. A primeira linha contém o valor de N ($1 \leq N \leq 10^5$). A segunda linha é composta por N inteiros A_i ($1 \leq A_i \leq 10^9$), que representam os valores dos elementos do array.

Output

Imprima, em uma linha, o caractere 'Y', caso o array de entrada seja especial, ou o caractere 'N', caso contrário.

Examples

standard input	standard output
3 1 2 1	Y
7 1 2 5 2 4 3 4	Y
6 1 2 3 1 2 3	N

Note

No terceiro exemplo, o próprio array da entrada é um subarray que não contém qualquer elemento que apareça uma única vez.

Problem G. Gole Perfeito

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Dani decidiu fazer o suco de limão perfeito. Ela acredita que o mais importante num suco de limão é o equilíbrio entre acidez e doçura. Para cada suco que ela faz, ela consegue medir quantas unidades de acidez e unidades de doçura o suco possui.

Após fazer N copos de suco de limão que não a agradaram, Dani encontrou uma receita secreta que afirma que o copo de suco perfeito deve ter exatamente A unidades de acidez e D unidades de doçura. O problema é que os limões acabaram e, como estamos no meio de uma pandemia, ela prefere não sair de casa para comprar mais limões. Logo, para criar o suco perfeito, só resta a ela misturar os sucos já feitos para conseguir chegar na medida ideal que caiba num copo de suco. **O copo final não precisa estar totalmente cheio**.

Em outras palavras, Dani deseja pegar uma fração (entre 0 e 1) de cada copo de suco para conseguir fazer um copo de suco final com A unidades de acidez e D unidades de doçura. Lembrando que ao pegar uma fração p de algum copo de suco i, que tem a unidades de acidez e d unidades doçura, este contribuirá com $p \cdot a$ de acidez e $p \cdot d$ de doçura no copo final.

Formalmente, sejam a_i e d_i as quantidades de acidez e doçura do copo i e seja p_i $(0 \leq p_i \leq 1)$ a fração do quanto o copo de suco i contribui para o copo final, então Dani quer que $\sum_{i=1}^{n} p_i \leq 1$ e $\sum_{i=1}^{n} p_i \cdot a_i = A$ e $\sum_{i=1}^{n} p_i \cdot d_i = D$.

Input

A primeira linha da entrada consiste de um inteiro T $(1 \leq T \leq 10^5)$.

A primeira linha de cada caso de teste consiste em 3 números n $(1 \leq n \leq 10^5)$, A e D $(0 \leq A, D \leq 10^9$ e $A + D > 0)$, representando o número de copos de sucos já feitos, a quantidade de acidez ideal e a quantidade de doçura do copo de suco perfeito, respectivamente. As próximas n linhas consistem, cada uma, em dois números a_i e d_i $(0 \leq a_i, d_i \leq 10^9)$, indicando quantas unidades de acidez e doçura o copo de suco i fornece.

É garantido que a soma de n para todos os casos de teste é, no máximo, 10^5.

Output

Para cada caso de teste. Caso não seja possível criar o copo de suco ideal, imprima uma linha com o caracter `N`. Caso contrário, imprima $n + 1$ linhas: a primeira com o caractere `Y` e as próximas n com os valores indicando a fração p_i $(0 \leq p_i \leq 1)$ que cada suco contribui para o copo final.

Sua resposta será considerada correta caso o erro absoluto ou relativo entre A e $\sum_{i=1}^{n} p_i \cdot a_i$ seja menor que 10^{-5} e o erro absoluto ou relativo entre D e $\sum_{i=1}^{n} p_i \cdot d_i$ seja menor que 10^{-5}.

Examples

standard input	standard output
1 4 4 3 2 2 2 5 5 2 5 5	Y 0.000000 0.000000 0.333337 0.466667
1 4 2 4 2 2 2 5 5 2 5 5	Y 0.000000 0.666668 0.000000 0.133333
1 4 3 1 2 2 2 5 5 2 5 5	N

Note

No primeiro caso, Dani pode misturar 0.33 do terceiro copo com 0.46 do quarto copo. O copo final terá $0.33 \cdot 5 + 0.46 \cdot 5 = 4$ de **acidez** e $0.33 \cdot 2 + 0.46 \cdot 5 = 3$ de **doçura**. Perceba que o copo final de suco estará apenas 80% cheio, o que está ok para Dani!

No terceiro caso, é impossível criar o copo de suco ideal.

Problem H. Helena e os Cheques

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Helena é uma prestadora de serviços que recebe, quase sempre, em cheques. Ela precisa pagar suas dívidas mensais, as quais totalizam, no mínimo, A reais. Dentre os N cheques que ela tem em seu poder, cada um no valor de x_i reais, ela deseja escolher alguns deles de modo que a soma dos valores dos cheques escolhidos totalize, no mínimo, A. Contudo, receando sair de casa com uma importância muito grande, ela também deseja que a soma dos valores dos cheques escolhidos não ultrapasse B reais.

Dados os valores de N, A, B e dos valores x_i de cada cheque, auxilie Helena determinando o número de maneiras distintas que os cheques escolhidos somem valores que estejam entre A e B, inclusive. Como este valor pode ser muito grande, imprima o resto de sua divisão por $10^9 + 7$.

Input

A primeira linha da entrada contém os valores de N, A e B ($1 \leq N \leq 100$, $1 \leq A \leq B \leq 2 \times 10^5$), separados por um espaço em branco.

A segunda linha contém N inteiros x_i ($1 \leq x_i \leq 10^5$, $1 \leq i \leq N$), separados por um espaço em branco.

Output

Imprima, em uma linha, o resto da divisão do número de maneiras distintas de se escolher cheques de modo que a soma dos cheques escolhidos esteja entre A e B, inclusive, por $10^9 + 7$.

Examples

standard input	standard output
3 5 10 3 2 3	4
5 10 10 1 2 3 4 5	3
10 1 350 12 25 77 89 31 67 55 98 10 2	914

Note

No primeiro caso, há quatro formas de se escolher cheques com valores entre 5 e 10 reais:

1. cheques 1 e 2 ($3 + 2 = 5$ reais)
2. cheques 1 e 3 ($3 + 3 = 6$ reais)
3. cheques 2 e 3 ($2 + 3 = 5$ reais)
4. cheques 1, 2 e 3 ($3 + 2 + 3 = 8$ reais)

No segundo caso, há três formas:

1. cheques 1, 2, 3 e 4 ($1 + 2 + 3 + 4 = 10$ reais)
2. cheques 1, 4 e 5 ($1 + 4 + 5 = 10$ reais)
3. cheques 2, 3 e 5 ($2 + 3 + 5 = 10$ reais)

Problem I. IMC aproximado

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

O índice de massa corpórea (IMC) é um dos indicadores de saúde de um indivíduo, computado a partir de sua massa m, em quilos, e sua altura h, em metros, por meio da relação

$$IMC = \frac{m}{h^2}$$

O intervalo $[18.5, 24.9]$ é considerado a faixa de normalidade: valores de IMC abaixo ou acima desta faixa correspondem a magreza ou a obesidade, respectivamente.

Uma recomendação informal é que, para estar dentro da faixa de normalidade, basta ao indivíduo ter massa igual ao descarte da parte inteira de sua altura. Por exemplo, se o indivíduo tem 1.70 m de altura, ele deveria ter massa igual a 70 kg.

Dada a altura do indivíduo, em metros, determine se esta aproximação leva ou não a um IMC dentro da faixa da normalidade.

Input

A entrada consiste em uma única linha contendo a altura h ($1.01 \leq h \leq 1.99$) do indivíduo, em metros. É garantido que a altura será dada com exatamente duas casas decimais.

Output

Se o IMC resultante da aproximação descrita estiver dentro da faixa de normalidade, imprima, em uma linha, a mensagem "Sim". Caso contrário, imprima "Nao" (sem acento e sem aspas).

Examples

standard input	standard output
1.70	Sim
1.92	Nao
1.16	Nao

Note

No primeiro caso, o IMC aproximado seria igual 24.22, dentro da faixa da normalidade.

No segundo caso, o valor aproximado do IMC está acima da faixa de normalidade (24.95).

No terceiro caso, a aproximação levaria à magreza (11.89).

Problem J. Jankenpon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Jankenpon, conhecido no Brasil como Pedra-Papel-Tesoura, é um jogo onde os participantes escolhem uma dentre três opções de jogada. As regras são as seguintes:

1. a pedra vence a tesoura e perde para o papel,
2. o papel vence a pedra e perde para a tesoura,
3. a tesoura vence o papel e perde para a pedra,
4. itens iguais empatam.

Quando há N participantes na partida, a pontuação do i-ésimo jogador é determinada pelo resultado do confronto de sua opção contra as opções escolhidas por todos os demais. Inicialmente todos os jogadores tem pontuação igual a zero. Cada vitória dá ao jogador um ponto, cada derrota subtrai um ponto e os empates são descartados. Os jogadores vitoriosos serão aqueles que obtiveram a maior pontuação dentre todos os jogadores.

Dado o valor de N e as opções de cada um dos jogadores, determine o total de vencedores.

Input

A primeira linha da entrada contém o valor de N ($2 \leq N \leq 2 \times 10^5$).

A segunda linha contém uma string s de tamanho N composta pelos caracteres '`R`', '`P`' ou '`S`', que correspondem à pedra (*rock*), papel (*paper*) e tesoura (*scissors*), respectivamente. O i-ésimo caractere de s indica a opção escolhida pelo i-ésimo jogador.

Output

Imprima, em uma linha, o número de jogadores vitoriosos.

Examples

standard input	standard output
2 RS	1
5 PRPRP	3

Note

No primeiro caso, o jogador 1 derrota o jogador 2, obtem um ponto e é o vencedor da disputa.

No segundo caso,

- O jogador 1 vence os jogadores 2 e 4, e empata com os jogadores 3 e 5, obtendo 2 pontos;
- O jogador 2 perde para os jogadores 1, 3 e 5 e empata com o jogador 4, finalizando a disputa com -3 pontos;
- O jogador 3 vence os jogadores 2 e 4, e empata com os jogadores 1 e 5, obtendo 2 pontos;
- O jogador 4 perde para os jogadores 1, 3 e 5 e empata com o jogador 2, finalizando a disputa com -3 pontos;

- O jogador 5 vence os jogadores 2 e 4, e empata com os jogadores 1 e 3, obtendo 2 pontos.

Portanto os jogadores 1, 3 e 5 são os vencedores, pois obtiveram a maior pontuação dentre todos os jogadores (2 pontos).

Problem K. Kaboom

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Kaboom! é um jogo de estratégia para dois jogadores que simula uma bomba a ser desarmada. A bomba é composta por N sensores, os quais estão conectados por M fios. Inicialmente, cada sensor está conectado, direta ou indiretamente, a todos os demais sensores. Caso um sensor se desconecte de um outro sensor, a bomba explodirá imediatamente.

Durante a partida os jogadores se alternam em turnos, e a cada turno o jogador deve cortar ou a ou b fios, mantendo os sensores conectados. O jogador que explodir a bomba será o perdedor. Ana e Beto disputarão uma partida, sendo que Ana iniciará o jogo.

Dados valores de N, a, b e as informações dos M fios, determine quem será o vencedor da partida, sabendo que ambos jogarão de forma ótima.

Input

A primeira linha da entrada contém os valores N ($2 \leq N \leq 10^5$), M ($N - 1 \leq M \leq 2 \times 10^5$), a e b ($1 \leq a < b \leq 100$), separados por um espaço em branco.

As M linhas seguintes contém os inteiros x_i e y_i ($1 \leq x_i, y_i \leq N, x_i \neq y_i$), separados por um espaço em branco, que indicam que o i-ésimo fio conecta os sensores x_i e y_i. É garantido que cada par de sensores só aparece no máximo uma vez e que todos os sensores estão conectados inicialmente.

Output

Imprima, em uma linha, o nome do vencedor da partida: "`Ana`" ou "`Beto`".

Examples

standard input	standard output
3 3 1 2 1 2 1 3 2 3	Ana
4 6 7 10 1 2 1 3 1 4 2 3 2 4 3 4	Beto

Note

No primeiro caso, Ana pode cortar qualquer um dos três fios, encerrando seu turno. Beto, por sua vez, ao cortar qualquer um dos fios restantes, explodirá a bomba, perdendo a partida.

No segundo caso, Ana teria que cortar no mínimo sete fios, sendo que há apenas seis. Deste modo, ela perde a partida, uma vez que os sensores se desconectariam no processo.

Problem L. Laços Infinitos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Marla é uma professora de Organização e Arquitetura de Computadores que resolveu solicitar aos seus alunos um interpretador simples de *assembly* ISPM (instruções simples da professora Marla) para os seus alunos. Neste *assembly*, os programas são escritos com uma instrução por linha, que são numeradas a partir de 0. O interpretador solicitado por Marla deve possuir acesso a 8 registradores, identificados de `R0` a `R7`, que devem ser inicializados com zero na execução de um programa. Considerando `RX`, `RY` e `RZ` três registradores, `INTEIRO` um número inteiro com sinal de 32 bits e `ENDERECO` um inteiro contendo o número da linha ocupada por uma instrução, temos que o conjunto de instruções que deve ser suportada pelo interpretador é o seguinte:

- `MOV RX INTEIRO`: atribui a `RX` o valor do `INTEIRO`
- `MOV RX RY`: atribui a `RX` o valor de `RY`.
- `ADD RX RY RZ`: soma os valores de `RY` e `RZ` e armazena o resultado em `RX`.
- `SUB RX RY RZ`: subtrai de `RY`, `RZ` e armazena o resultado em `RX`.
- `MUL RX RY RZ`: multiplica `RY` por `RZ` e armazena o resultado em `RX`.
- `DIV RX RY RZ`: divide `RY` por `RZ` e armazena o resultado em `RX`. A divisão é inteira, isto é, a parte fracionária é desprezada.
- `MOD RX RY RZ`: toma o resultado de `RY` *mod* `RZ` e armazena o resultado em `RX`.
- `BEQ RX RY ENDERECO`: o fluxo do programa é direcionado para a instrução que ocupa a linha de número `ENDERECO` caso `RX` seja igual a `RY`.
- `BLT RX RY ENDERECO`: o fluxo do programa é direcionado para a instrução que ocupa a linha de número `ENDERECO` caso `RX` seja menor que `RY`.
- `JMP ENDERECO`: o fluxo do programa é direcionado para a instrução que ocupa a linha de número `ENDERECO`.
- `PRINT RX`: o valor de `RX` é impresso na tela.
- `EXIT`: encerra o programa.

Como se trata de uma linguagem brinquedo e os interpretadores desenvolvidos não executarão problemas muito complexos, Marla solicitou que, caso o programa levasse 10^5 ou mais instruções sem achar uma instrução `EXIT`, o interpretador deveria acusar um laço infinito e interromper a execução do programa. Implemente o interpretador que Marla pediu, caso contrário você ficará pendurado em OAC!

Input

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 50$) indicando o número de instruções do programa. As próximas N linhas possuem cada, uma instrução válida.

É garantido que existe exatamente uma instrução `EXIT` em um programa bem formato de Assembly ISPM.

Output

A cada comando de `PRINT RX` o interpretador deve imprimir em uma linha o valor do registrador `RX`. Caso o programa leve mais de 10^5 instruções para finalizar, o interpretador deverá imprimir em uma linha a mensagem "`laco infinito!`" e encerrar o programa.

Examples

standard input	standard output
5 MOV R0 1 MOV R1 1 ADD R0 R1 R1 PRINT R0 EXIT	2
7 MOV R0 0 MOV R1 100000 MOV R2 1 BEQ R0 R1 6 ADD R0 R0 R2 JMP 3 EXIT	laco infinito!
8 MOV R0 5 MOV R1 3 BLT R0 R1 5 MOV R2 R1 JMP 6 MOV R2 R0 PRINT R2 EXIT	3

Note

O primeiro exemplo corresponde a um programa que soma $1 + 1$ e deixa 2 como resposta.

O segundo exemplo é programa contendo um laço de repetição que vai de 0 à $10^5 - 1$. Como ele requer mais do que 10^5 instruções para ser finalizado, a mensagem "`laco infinito!`" é impressa na tela.

O terceiro exemplo pega o mínimo entre os valores contidos nos registradores `R0` e `R1` e imprime este valor na tela.

Problem M. Média Móvel

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Uma média móvel é um estimador calculado a partir de amostras sequenciais da população. Dada uma sequência de valores, o primeiro elemento em uma média móvel é a média dos primeiros n valores desta sequência. Médias móveis são comumente usadas com séries temporais para suavizar flutuações curtas e destacar tendências de longo prazo.

A média móvel simples sobre uma sequência $P = (p_1, \ldots, p_m)$ é a sequência $\overline{P} = (\overline{p}_1, \ldots, \overline{p}_{m-n+1})$ das médias das subsequências de n elementos consecutivos de P. Ou seja, dada a sequência $P = (p_1, \ldots, p_m)$, o cálculo de um termo qualquer da sequência resultante pela média móvel é dado por:

$$\overline{p}_i = \frac{1}{n} \sum_{j=0}^{n-1} p_{i+j}$$

Dados os valores de m e n, calcule a média móvel simples de uma sequência.

Input

A primeira linha da entrada consiste de dois valores inteiros m e n, separados por espaço, tais que $1 \leq n < m \leq 10^6$. A segunda linha apresenta a sequência de valores inteiros $-10^9 \leq p_i \leq 10^9$, $1 \leq i \leq m$.

Output

Apresente a sequência de valores para a média móvel simples, um valor real por linha. Sua resposta será considerada correta se o erro relativo ou absoluto não exceder 10^{-6}, ou seja, sendo sua resposta a e a do juiz b, se $\frac{|a-b|}{\max(1,b)} \leq 10^{-6}$.

Examples

standard input	standard output
10 2 1 2 3 4 5 6 7 8 9 10	1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5
8 5 1024 999 898 1047 1560 1300 1420 1000	1105.6 1160.8 1245.0 1265.4

Note

No primeiro caso, os dados são considerados dois a dois, e a média móvel apresenta uma clara tendência crescente.

Problem N. Natal e árvores

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

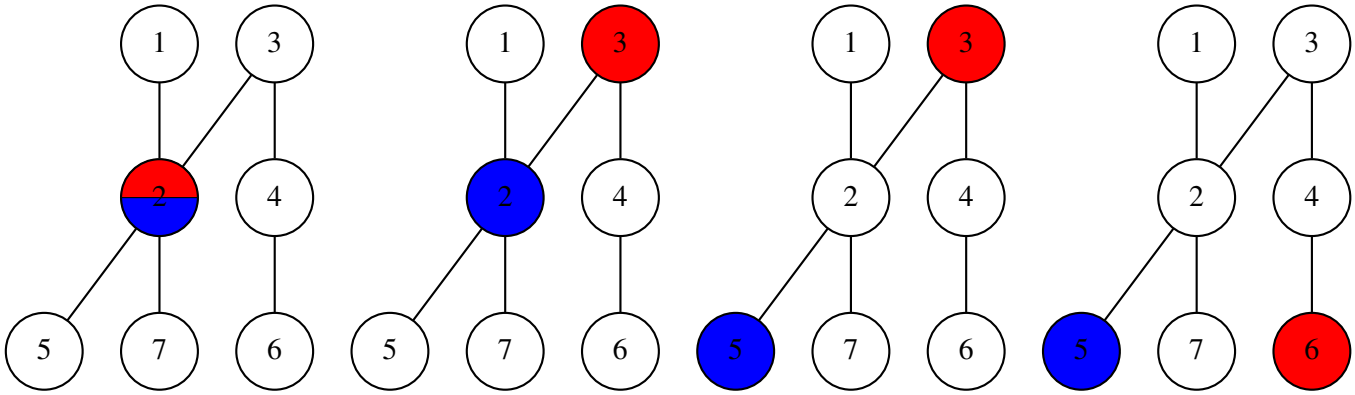
Em épocas natalinas árvores ficam à tona. Na teoria dos grafos, árvore é um grafo com N vértices, $N-1$ arestas e sem ciclos.

Amanda e Bianca bolaram um jogo em árvore. Cada jogadora tem uma peça que só ela pode mexer. Inicialmente ambas as peças são colocadas no mesmo vértice e as jogadoras movem alternadamente. Quem faz o primeiro movimento é decidido num par ou ímpar. Um *movimento* é composto de:

- A jogadora começa a mover sua peça de forma que a distância entre as peças sempre aumenta. Caso haja várias formas de fazê-lo, a jogadora pode escolher qualquer uma.
- Assim que a peça chega num vértice com identificador maior que o do vértice com a peça da oponente, o turno acaba.

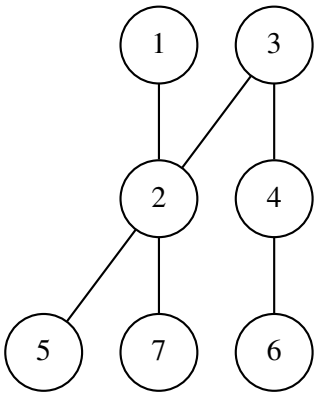
Caso uma jogadora não consiga terminar um movimento em seu turno, ela perde.

A figura abaixo ilustra um jogo possível onde a peça de Amanda é representada pela cor vermelha e a de Bianca pela cor azul. O jogo começa no vértice 2 e Amanda ganha o par ou ímpar. Amanda move para 3. Bianca move para 5. Amanda move para 4, mas não pode parar lá e continua para o vértice 6.



Podemos definir o estado de um jogo, que não está no meio de um movimento, como um par (a, b) onde a é o vértice com a peça de Amanda e b é o vértice com a peça de Bianca.

Na árvore da figura abaixo, alguns dos estados possíveis são: $(1, 2)$, $(3, 3)$, $(4, 7)$, $(6, 5)$, $(5, 6)$. Alguns dos estados não possíveis são: $(2, 4)$, $(7, 1)$, $(1, 7)$, $(6, 3)$.



Clara gosta de combinatória e se perguntou quantos estados são possíveis de serem alcançados para uma determinada árvore sem saber quem ganha no par ou ímpar. Ajude Clara a calcular este valor!

Input

A primeira linha da entrada terá um inteiro T ($1 \leq T \leq 10^5$) indicando o número de casos de testes.

A primeira linha de cada caso de teste terá um inteiro N ($1 \leq N \leq 2 \cdot 10^5$), a quantidade de vértices na árvore.

As próximas $N - 1$ linhas contém, cada uma, dois inteiros u e v ($1 \leq u, v \leq N$) as arestas da árvore. É garantido que as arestas informadas formam uma árvore.

É garantido que a soma de N para todos os casos de teste será, no máximo, $5 \cdot 10^5$.

Output

Para cada caso de teste, imprima a quantidade de configurações possíveis.

Example

standard input	standard output
3 3 1 2 2 3 7 1 2 2 3 3 4 2 5 4 6 2 7 5 1 2 2 3 3 4 1 5	7 33 19

Note

No primeiro caso de teste os estados válidos são: $(1, 1)$, $(1, 2)$, $(2, 1)$, $(2, 2)$, $(2, 3)$, $(3, 2)$, $(3, 3)$.

Problem O. Observação Estelar

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Nélio das Graças estuda astronomia por *hobby*, seu passatempo é observar o céu estrelado e tentar identificar as constelações. Como ainda é inexperiente, Nélio procura se localizar no céu estrelado através de estrelas de padrão bem conhecido, como a constelação do "Cruzeiro do Sul" ou o asterismo das "Três Marias".

Como Nélio pretende viajar para o deserto do Atacama para conseguir observar as estrelas com nitidez, ele precisará, devido à quantidade de estrelas, de ajuda para localizar a constelação de referência, que a guiará no estudo dos astros. Assim, Nélio pretende tirar uma foto do céu estrelado e enviá-la à você para que, juntamente com uma imagem da constelação de referência, consiga informar a Nélio onde tal constelação se localiza na foto.

Considere que as linhas e as colunas possuem numeração iniciando de 0, portanto, a linha do topo e a coluna mais à esquerda possuem índice 0.

Input

A primeira linha da entrada possui 4 inteiros N, M ($1 \leq N, M \leq 2000$), K e L ($1 \leq L, K \leq 60$), separados por um espaço em branco, os quais indicam, respectivamente, as dimensões da fotografia do céu estrelado e as dimensões da imagem com a constelação de referência.

As próximas N linhas contém, cada uma, uma string de tamanho M formada pelos caracteres $\{*, .\}$, indicando a composição de cada linha da fotografia.

As próximas K linhas contém, cada uma, uma string de tamanho L formada pelos caracteres $\{*, .\}$, correspondendo a composição de cada linha da imagem da constelação de referência.

Output

Imprima, em uma linha, dois inteiros x e y, separados por um espaço em branco, as quais representam as coordenadas de uma ocorrência da constelação de referência na foto do céu estrelado. Caso a constelação de referência não ocorra na imagem, imprima "`-1 -1`".

Examples

standard input	standard output
4 4 2 2 .... .*.. .*.. .... *. *.	1 1
4 4 2 1 .*.* *.*. .*.* *.*. * .	0 1
5 5 2 2 ..... .***. .***. .***. ..... ** **	1 1

Note

No primeiro exemplo, a constelação de referência ocorre na fotografia a partir da coordenada $(1, 1)$.

O segundo caso é análogo.

No terceiro exemplo, a constelação de referência ocorre na fotografia partir das coordenadas $(1, 1)$, $(1, 2)$, $(2, 1)$ e $(2, 2)$. Note que as ocorrências se sobrepõem.

Problem P. Pamonhas

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Além de ser responsável por ofertar cursos de graduação, a Faculdade do Gama se especializou em fazer pamonhas de excelente qualidade graças aos dotes culinários dos professores Edson Alves, John Lennon e Bruno Ribas. Atualmente, a FGA é capaz de produzir D pamonhas doces, S pamonhas salgadas e F pamonhas *fit* por dia.

Uma ação beneficente proposta pelos professores Daniel Saad, Guilherme Ramos e Roberta Barbosa consiste em juntar todas as pamonhas produzidas em um dia e organizá-las em kits para serem entregues às instituições de caridade. Os professores organizam esses kits de forma que cada um tenha pamonhas de um único tipo e em mesma quantidade, que também deve ser o maior número possível dos deliciosos quitutes.

Como os professores estão preocupados com o tempo de preparação e embalagem, eles pedem sua ajuda. Sua tarefa consiste em determinar a quantidade máxima de pamonhas de cada tipo que podem estar em cada kit.

Input

A entrada consiste de uma linha contendo três números inteiros, separados por espaço, D, S e F ($1 \leq D, S, F \leq 10^9$), representando a quantidade de cada tipo de pamonhas, doces, salgadas e *fit*, respectivamente.

Output

Imprima, em uma linha, a quantidade máxima de cada tipo de pamonha que compõe o kit, levando em conta que cada kit deve possuir a mesma quantidade de pamonhas e todas as pamonhas serão usadas para montar kits.

Examples

standard input	standard output
4 8 6	2
3 1 5	1
24 12 16	4
35 20 15	5

Note

No primeiro exemplo de teste, podemos ter kits de tamanho 1 (somando 18 kits) ou kits de tamanho 2 (somando $2+2$, $2+2+2$, $2+2+2+2 = 9$ kits) e nenhum outro tamanho é valido. A resposta para o problema é 2.

No segundo caso de teste, como temos apenas 1 pamonha de sal, podemos montar kits com todas as pamonhas, sendo cada kit com pamonhas de mesmo tipo, com apenas uma única pamonha cada.

No terceiro caso, os tamanhos possíveis para os kits são: todos com 1, todos com 2 ou todos com 4. Portanto, a quantidade máxima é 4.