2019
Caderno de Problemas
Maratona Mineira de Programação

Patrocínio

Accenture Applied Intelligence

Algar Telecom

NEPRE CONSULTING
SISTEMAS INDUSTRIAIS

tfg

Calaboração

FUNDAÇÃO THEODOMIRO SANTIAGO
RES, NON VERBA

Realização

UNIFENAS

Informações gerais

Este caderno de tarefas é composto por **??** páginas (não contando a folha de rosto), numeradas de 1 a **??**. Verifique se o caderno está completo.

Nome do programa

Cada problema tem os possíveis nomes de arquivo fonte indicados abaixo do título. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python devem ser arquivos com sufixo *.py*.

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada consiste em exatamente um caso de teste, que é descrito usando uma quantidade de linhas que depende do problema. O formato da entrada é como descrito em cada problema. A entrada não contém nenhum conteúdo extra.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim de linha (\n).
- A entrada não contém linhas vazias.
- Quando a entrada contém múltiplos valores separados por espaços, existe exatamente um espaço em branco entre dois valores consecutivos na mesma linha.

Saída

- A saída deve ser escrita na saída padrão.
- A saída deve respeitar o formato especificado no enunciado. A saída não deve conter nenhum dado extra.
- Todas as linhas da saída, incluindo a última, devem terminar com o caractere de fim de linha (\n).
- Quando uma linha da saída apresentar múltiplos valores separados por espaços, deve haver exatamente um espaço em branco entre dois valores consecutivos.
- Quando a um valor da saída for um número real, use pelo menos o número de casas decimais correspondente à precisão requisitada no enunciado.

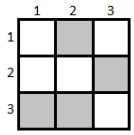
Problema A. Navalha Batal

Nome do arquivo fonte: `batal.c`, `batal.cpp`, *ou* `batal.java`

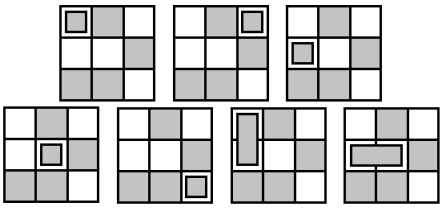
Bibika está jogando o famoso jogo Navalha Batal. Para quem não conhece, o jogo é disputado entre duas pessoas e um tabuleiro comum, de tamanho $N \times N$, com algumas peças (que representam Bavios) de tamanhos $1 \times T$ e $T \times 1$ previamente inseridas, onde o valor de T é inteiro positivo menor ou igual a N. A sobreposição de peças não é possível!

Após a posição das peças iniciais ser revelada, cada jogador tem alguns minutos para analisar o tabuleiro e calcular (ou chutar) a quantidade de peças $1 \times T$, $T \times 1$ diferentes que ainda podem ser inseridas no tabuleiro. Duas peças são diferentes se os espaços que elas ocupam no tabuleiro são diferentes.

Segue um exemplo de um tabuleiro 3×3 com as posições $\{(1,2),(2,3),(3,1),(3,2)\}$ previamente preenchidas.



Nesse caso ainda é possível inserir nas células vazias 7 peças diferentes, como ilustrado abaixo:



Após jogarem com um tabuleiro bem pequeno, Bibika gostaria de saber, dado um tabuleiro gigante e uma configuração inicial do tabuleiro, qual é a solução do jogo. Como é uma tarefa bastante complexa a olho nu, cabe a você ajudá-la!

Entrada

A primeira linha contém dois inteiros N e Q, sendo N o número de linhas e colunas do tabuleiro e Q a quantidade de células distintas que estão previamente preenchidas. As próximas Q linhas possuem dois inteiros, X_i e Y_i, indicando que a coordenada X_i, Y_i do tabuleiro está preenchida.

Saída

A saída deve conter uma linha com um único inteiro representando a quantidade de bavios de tamanho $1 \times T$ ou $T \times 1$ que ainda são possíveis de serem colocados de forma que fiquem totalmente inseridos no tabuleiro e não exista sobreposição com outros bavios.

Restrições

- $1 \leq N \leq 10^6$
- $0 \leq Q \leq 10^5$
- $1 \leq X_i, Y_i \leq N$, para todo $i = 1, 2, \ldots, Q$

Exemplos

Entrada	**Saída**
3 4 1 2 2 3 3 1 3 2	7

Entrada	**Saída**
2 3 1 1 1 2 2 2	1

Problema B. Podando Galhos

Nome do arquivo fonte: `galhos.c`, `galhos.cpp`, *ou* `galhos.java`

O natal já passou. Porém, por causa da correria do dia-a-dia, a árvore de natal na casa de Graça continua montada. Mas a família dessa garota-prodígio decidiu fazer um mutirão no próximo fim de semana para desmontar a árvore, onde também ainda está a toca do coelhinho da páscoa.

Como Ada, a mãe de Graça, é uma cientista da computação apaixonada pelo que faz, ela nunca perde uma oportunidade de introduzir a filha no fabuloso mundo dos algoritmos e estruturas de dados. Por isso, a árvore de natal da família foi realmente montada como uma árvore que você encontraria em um curso de algoritmos: ela possui bolas numeradas de 1 a N, cada uma podendo ser vista como um vértice, pendurada a outra bola acima dela, com exceção da raiz que foi pendurada no teto.

Para que o desmonte da árvore seja mais divertido, Ada propôs um desafio a Graça. No i-ésimo segundo do desmonte, Ada irá cortar o fio que originalmente mantinha a bola de número B_i suspensa. Caso a bola ainda esteja suspensa, ela irá então cair ao chão, juntamente com todas as bolas que estavam penduradas nela direta ou indiretamente. Pode acontecer que Ada corte o fio de uma bola que já esteja no chão - neste caso, nada acontece. Passados N segundos, com todas as bolas no chão, Ada pergunta a Graça em qual segundo cada uma das N bolas caiu. Caso Graça acerte todos os números, ela será recompensada com uma viagem para assistir a próxima edição da Maratona Mineira, onde ela poderá conhecer outras jovens mentes promissoras como a sua!

Graça é muito ágil e conseguiu anotar todos os números necessários manualmente. Mas será que você consegue fazer um programa para ajudá-la a se preparar para o desmonte de árvore de natal do ano que vem? Considere que as bolas caem imediatamente ao chão quando deixam de estar presas direta ou indiretamente ao teto.

Entrada

A primeira linha da entrada contém um inteiro N, o número de bolas da árvore de natal. A segunda linha contém N inteiros $P_1, \cdots, P_N$ separados por espaço. P_i indica que há um fio prendendo a bola i à bola P_i, que está acima na árvore. $P_i = -1$ indica que a bola i é a raiz da árvore. Há sempre exatamente uma bola i com $P_i = -1$, e é garantido que as ligações dadas descrevem uma árvore. A última linha contém N inteiros $B_1, \cdots, B_N$ separados por espaço, indicando qual fio é cortado no i-ésimo segundo do processo de desmonte. Cada inteiro entre 1 e N aparece exatamente uma vez na última linha da entrada.

Saída

Escreva na saída uma linha com N inteiros separados por espaço. O i-ésimo desses inteiros deve indicar em qual segundo a bola i cai ao chão.

Restrições

- $1 \leq N \leq 2 \times 10^5$

Exemplos

Entrada	**Saída**
2 2 -1 2 1	1 1

Entrada	**Saída**
5 4 4 5 -1 4 2 1 5 4 3	2 1 3 4 3

Entrada	**Saída**
9 9 8 8 6 8 2 6 -1 5 1 7 5 6 2 9 4 3 8	1 5 8 4 3 4 2 9 3

Problema C. Caça às Flores

Nome do arquivo fonte: `flores.c`, `flores.cpp`, *ou* `flores.java`

Damiko precisa fazer uma pesquisa sobre T diferentes tipos de flores para a escola. Para isso ele irá usar o jardim existente em sua casa para colher esses T tipo de flores. Como ele deixou para realizar a pesquisa de última hora, precisa colher as flores de forma rápida e portanto ele necessita de sua ajuda para encontrar a menor área quadrada do jardim que contém todos os tipos de flores (1 a T inclusive).

Sua tarefa é simples. Damiko irá fornecer um mapa do seu jardim no formato de uma matriz $N \times M$ onde, em cada posição, há um valor inteiro representando os tipos de flores no local. O i-ésimo bit menos significativo desse inteiro é 1 se e somente se nessa posição da matriz existe a flor do tipo i. Portanto se uma posição do jardim contém, por exemplo, o valor 5, isso significa que existem dois tipos de flores: o tipo 1 e o tipo 3. Caso o valor seja 7 ela contém os tipos: 1, 2 e 3. Caso seja 0, não existem flores no local.

Segue um exemplo de mapa 3×4 de um jardim com $T = 5$ tipos de flores:

	0	1	2	3
0	0	1	9	3
1	2	1	0	8
2	3	0	7	20

No exemplo do jardim acima a menor região que contém os 5 tipos diferentes de flores é a de área 4 destacada iniciando na posição $(1, 2)$ e finalizando na posição $(2, 3)$.

Entrada

A primeira linha da entrada contém três inteiros N, M e T separados por espaço, onde N e M são as dimensões do jardim e T é o número de tipos flores a serem colhidas. Cada uma das N linhas seguintes contém M inteiros separados por espaço. O j-ésimo inteiro da i-ésima dessas linhas, X_{ij}, representa as flores na posição $(i - 1, j - 1)$ do jardim.

Saída

Caso seja possível coletar todas as flores colhendo uma área quadrada do jardim, a primeira linha da entrada deve conter um inteiro indicando a menor área quadrada que deve ser coletada. A segunda linha deve conter dois inteiros L e C separados por espaço, indicando respectivamente a linha e a coluna do canto superior esquerdo do quadrado coletado. Caso haja mais de uma resposta possível, escolha uma que minimize L. Se, mesmo assim, ainda houver mais de uma resposta possível, escolha aquela que minimiza C. Caso não seja possível coletar todas as flores colhendo uma área quadrada do jardim, a saída deve conter uma linha com o inteiro -1.

Restrições

- $1 \leq N, M \leq 1000$
- $1 \leq T \leq 10$
- $0 \leq X_{ij} \leq 1023$ para todo $i = 0, 1, \ldots, N-1$, $j = 0, 1, \ldots, M-1$.

Exemplos

Entrada	**Saída**
3 4 5 0 1 9 3 2 1 0 8 3 0 7 20	4 1 2

Entrada	**Saída**
2 2 1 2 4 6 0	-1

Entrada	**Saída**
2 3 1 4 2 3 2 3 1	1 0 2

Problema D. Prêmios da Mineira

Nome do arquivo fonte: `premios.c`, `premios.cpp`, *ou* `premios.java`

A Maratona Mineira de Programação é um evento de muita consagração e sucesso. Há oito anos, a Maratona Mineira vem incentivando jovens estudantes a praticar programação competitiva, ajudando alunos de toda Minas Gerais a desenvolver habilidades lógicas e técnicas.

Em 2012, a primeira Maratona Mineira de Programação foi realizada em Uberlândia, e os primeiros colocados receberam prêmios muito legais! A organização deu o prêmio mais caro para o primeiro lugar, o segundo mais caro para o segundo lugar e assim por diante. Porém, dinheiro não é tudo nessa vida e Gabriel Prosa sabe disso. O time dele ficou em primeiro lugar e recebeu o prêmio mais caro. Porém, para ele, o prêmio mais legal de todos foi o dado ao K-ésimo time mais bem colocado.

Na Maratona de Programação temos N times competindo, cada time i resolveu P_i problemas com uma penalidade de T_i minutos. O ranking final é feito ordenando os times por ordem decrescente de problemas resolvidos, caso haja dois times com mesmo número de problemas resolvidos, o time com menor penalidade de tempo fica acima no ranking. É garantido que não temos dois times com o mesmo número de problemas e com a mesma penalidade.

Dado o número de problemas e a penalidade de cada time, qual será o time que ganhou o prêmio mais legal da competição, segundo Gabriel Prosa?

Entrada

A primeira linha da entrada é composta por dois inteiros N e K, representando o número de times que participaram da competição, e a posição do time que ganhou o prêmio mais legal, respectivamente. Logo após, teremos N linhas. A i-ésima dessas linhas é composta por três inteiros, ID_i, P_i e T_i, representando o identificador único do i-ésimo time, o número de problemas que o time resolveu, e a penalidade.

Saída

A saída deverá ser composta por somente um inteiro, o identificador do K-ésimo time mais bem colocado.

Restrições

- $1 \leq N \leq 100$
- $1 \leq K \leq N$
- $1 \leq ID_i \leq N$
- $1 \leq P_i \leq 13$
- $1 \leq T_i \leq 2000$

Exemplos

Entrada	**Saída**
5 3 2 10 720 3 7 120 1 3 135 5 10 563 4 11 735	2

Problema E. Jantar das Capivaras

Nome do arquivo fonte: `jantar.c`, `jantar.cpp`, *ou* `jantar.java`

Há vários anos, Bacon – o Bravo – saiu vitorioso das disputas de sucessão do reino das capivaras. Infelizmente, Bacon já está com 16 anos, uma idade avançada para uma capivara. Como todo bom rei absolutista, Bacon – o Bravo – também gostaria de ser conhecido como fundador da maior dinastia já vista por sua espécie, e almeja colocar no trono sua filha mais velha, Bacon – a Ambiciosa.

Para evitar um novo conflito entre seus vassalos, o monarca convocou todos os nobres de seu reino para um jantar onde será decidido o futuro monarca das capivaras, onde os convidados serão dispostos em uma grande mesa retangular com a família real ocupando as duas cabeceiras. Para deixar os nobres mais confortáveis, e assim aumentar suas chances de realizar seus sonhos, Bacon – o Bravo – quer dividir as nobre capivaras nas duas laterais da mesa tal que cada capivara tenha, no máximo, um de seus aliádos políticos do outro lado. Infelizmente, ou felizmente para nosso monarca, as richas entre os nobres ainda estão muito fortes e evidentes, e com isso cada nobre tem não mais do que três aliados. Os lados da mesa não precisam ter o mesmo número de ocupantes, mas cada um deve ter pelo menos uma capivara.

Ajude o rei a determinar se é possível ou não dividir os nobres dessa forma e seja imortalizado nas *Crônicas das Capivaras* como Humana/Humano – a/o Ajudante!

Entrada

A primeira linha da entrada contém dois inteiros separados por espaços, N e M, o número de nobres e o número de aliaças políticas existentes entre eles. Seguem M linhas, cada uma com dois inteiros u_i e v_i, indicando que a capivara u_i aliada à capivara v_i. Estamos falando de um reino, então não existem capivaras isoladas politicamente: sempre é possível chegar de uma capivara a outra apenas seguindo os elos de lealdade.

Saída

A primeira linha da saída deve conter um único inteiro R, que deve ser 0 caso seja impossível realizar a divisão, ou 1, casa seja possível. Se $R = 0$, não há mais linhas na saída. Se $R = 1$, a segunda linha contem um único inteiro S_1, o número de capivaras do lado esquerdo da mesa. Seguem então S_1 inteiros na terceira linha, separados por espaço, cada um representando o número de uma capivara que está do lado esquerdo. A quarta linha contém com um único inteiro S_2, o número de capivaras do lado direito da mesa. Seguem então S_2 inteiros na quinta linha, separados por espaço, cada um representando o número de uma capivara que está do lado direito.

Restrições

- $2 \leq N \leq 10^6$
- $1 \leq M \leq \frac{3N}{2}$
- $1 \leq u_i, v_i \leq N$

Exemplos

Entrada	**Saída**
3 3 1 2 2 3 1 3	0

Entrada	**Saída**
6 7 1 2 1 3 1 4 3 5 4 5 5 6 2 6	1 2 2 6 4 1 3 4 5

Problema F. Quem é o mentiroso?

Nome do arquivo fonte: `mentiroso.c`, `mentiroso.cpp`, *ou* `mentiroso.java`

Todo fim de ano, Marcos vai para Furnas comemorar o Natal junto com seus amigos. Eles tem a tradição de sempre jogar *Amigo Oculto* (também conhecido como *Amigo Secreto*) nas vésperas de natal. Para quem não sabe, Amigo Oculto é uma brincadeira na qual é sorteado para cada participante um outro participante o qual ele deve dar um presente, de forma que cada um deles também receba um presente e que ninguém tenha que dar presente para si mesmo.

Os amigos de Marcos inovaram e esse ano irão fazer uma surpresa para ele. Eles decidiram que um deles irá mentir sobre quem sorteou. Marcos não participará do sorteio mas deverá descobrir *quem é o mentiroso*! Se Marcos acertar ele ficará com os presentes de todos eles. Se errar deverá dar um presente para cada um!

Com muito medo de errar, Marcos pediu para que fizesse um programa que dado quem cada participante diz ter tirado, diga quem é o mentiroso. Se isso não for possível, diga quais participantes podem estar mentindo, pois afinal, qualquer dica será muito preciosa para Marcos!

Entrada

A entrada contém um número inteiro N que representa o número de pessoas que estão participando do Amigo Oculto. A i-ésima das próximas N linhas contém dois nomes separados por espaço, a_i e b_i, indicando que a pessoa a_i sorteou a pessoa b_i. Nenhum nome é muito esquisito, tendo no máximo 10 letras.

Saída

A primeira linha da saída deve conter um inteiro M representando o número de possíveis mentirosos. Cada uma das próximas M linhas deve conter o nome de um deles. Apresente os nomes em ordem alfabética.

Restrições

- $2 \leq N \leq 10^5$.
- $1 \leq |a_i|, |b_i| \leq 10$, para todo $i = 1, 2, \ldots, N$.
- Nomes contém apenas letras minúsculas.
- Os nomes dos N participantes são distintos.

Exemplos

Entrada	**Saída**
4 papa paulo nicao renan renan paulo paulo papa	2 papa renan

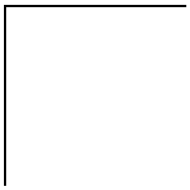
Problema G. Brincando com a formiga

Nome do arquivo fonte: `formiga.c`, `formiga.cpp`, *ou* `formiga.java`

Júlia gosta muito de animais domésticos. Ela tem vários cachorros, gatos, papagaios, mas seu animal preferido mesmo é sua formiga, que recebeu o nome de Fininha.

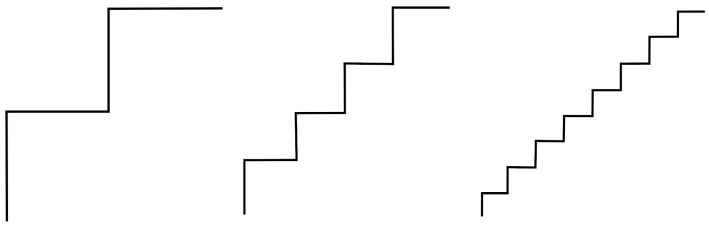
Aproveitando suas aulas de carpintaria, Júlia está construindo um brinquedo para Fininha.

Ela começou pegando dois pedaços de madeira de 1 metro cada um e construindo um degrau, como mostra a figura abaixo:



Fininha sempre começa do ponto mais baixo do degrau, e sobe pelos segmentos do brinquedo até chegar ao topo. Mas ela não para por aí: ao chegar ao topo, Fininha começa a descer, depois sobe novamente, e assim segue, até ter fome. Nesse momento, a formiga fica sem energia e precisa se alimentar. Júlia, como boa dona, pega então alguns grãos de açúcar e leva até a boca de seu animal de estimação.

Para que o brinquedo fique menos monótono, Júlia decidiu transformar cada segmento do degrau em dois com metade do tamanho. Ela repetiu esse procedimento K vezes. Cada vez que Júlia faz isso, o número de degraus do brinquedo dobra, e o tamanho de cada um dos novos segmentos fica exatamente metade do tamanho dos segmentos anteriormente. Na figura abaixo, você pode ver como o brinquedo ficou depois de uma, duas e três repetições desse procedimento.



Júlia notou que sua formiga demora 2 segundos para subir um segmento qualquer, independente do seu tamanho. Ela leva 1 segundo para percorrer um segmento horizontal, e apenas meio segundo para descer um segmento, qualquer que seja o seu tamanho. A formiga já está brincando há N segundos, e é hora de dar comida para ela. Mas Júlia está sem óculos, e precisa de você para alimentar Fininha. Você pode dizer exatamente onde está a formiga agora?

Entrada

A entrada contém apenas uma linha com os dois inteiros K e N, separados por espaço, representando o número de transformações feitas por Júlia no brinquedo e há quantos segundos Fininha já está brincando.

Saída

A primeira linha da saída deve conter dois números racionais separados por espaço, representando as coordenadas x e y da posição de Fininha no segundo N. Considere que o ponto mais baixo do brinquedo posicionado está na origem (ponto $(0, 0)$), e que o brinquedo termina na posição $(1, 1)$, onde Fininha começa a descer.

Essas posições não mudam com as transformações feitas. As coordenadas inteiras devem ser escritas como inteiros, enquanto as não inteiras devem ser escritas como uma fração irredutível da forma a/b.

Restrições

- $0 \leq K \leq 40$
- $0 \leq N \leq 10^{18}$

Exemplos

Entrada	**Saída**
`0 3`	`1 1`

Entrada	**Saída**
`1 10`	`0 1/4`

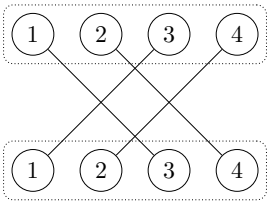
Entrada	**Saída**
`4 59`	`17/32 9/16`

Problema H. Comunicação Trocada

Nome do arquivo fonte: `comunicacao.c`, `comunicacao.cpp`, *ou* `comunicacao.java`

Luci visitou recentemente o Museu da História da Computação, onde teve a oportunidade de apreciar várias exposições. Em uma delas, conheceu várias calculadoras antigas como a *Step Reckoner*, criada por volta de 1672. Em outra, viu supercomputadores do passado, como os famosos Cray. Também conheceu várias maneiras de programar computadores antigos, como cartões perfurados e painéis de tomadas.

Um painel de tomadas em particular chamou a atenção de Luci. Nele, há dois blocos com N tomadas cada. Em cada bloco, as tomadas são numeradas de 1 a N, da esquerda para a direita. Usando uma tecnologia peculiar de cabos, o operador conectava as tomadas de um dos blocos às tomadas do outro bloco. Veja o exemplo com 4 tomadas em cada bloco na figura abaixo.



Em cada ciclo do computador ao qual esse painel pertence, cada tomada tem um sinal elétrico que representa um bit. O propósito desse painel é trocar a ordem dos bits. Por isso, cada tomada de cada bloco deve ser conectada a exatamente uma tomada do outro bloco. Cada conexão deve ser feita com um segmento de cabo ligando as duas tomadas em linha reta. Por restrições físicas relacionadas à tecnologia peculiar dos cabos, sempre que dois cabos se cruzam o bit transmitido em ambos os cabos muda de valor. Ou seja, se o bit 1 estava sendo transmitido por um cabo, este cabo agora passa a transmitir o bit 0. No exemplo da figura acima, o cabo conectando a tomada 1 do bloco superior à tomada 3 do bloco inferior intercepta dois cabos. Dessa forma, o bit transmitido por ele será invertido duas vezes, fazendo com que o bit na tomada 1 do bloco superior seja o mesmo que o na tomada 3 do bloco inferior.

Como o objetivo do painel é trocar a ordem dos bits, maneiras de conectar as tomadas que fazem com que o bits em duas tomadas conectadas sejam diferentes não são válidas. Ao perceber isso, Luci começou a tentar descobrir de quantas formas era possível conectar as tomadas de maneira válida. Ao perceber que o museu tinha vários painéis desses e que, em alguns deles, algumas conexões já haviam sido feitas, ela achou melhor aproveitar o restante da visita ao museu e deixou a pergunta para você: de quantas formas válidas é possível conectar as tomadas, preservando as conexões já existentes? Lembrando que uma forma é válida se cada tomada estiver conectada a exatamente uma tomada do bloco oposto e se o bit em tomadas conectadas for o mesmo.

Observações

- Se mais de dois cabos se cruzarem em um mesmo ponto, todos os cruzamentos par a par devem ser considerados.

Entrada

A primeira linha da entrada contém dois inteiros N e M, representando o número de tomadas em cada bloco e o número de conexões já existentes respectivamente.

Seguem M linhas, cada uma contendo dois inteiros a_i e b_i, indicando que a tomada a_i do bloco superior está conectada à tomada b_i do bloco inferior.

Saída

Se X é o número de formas válidas de conectar as tomadas preservando as conexões já existentes, a saída deve conter um único inteiro representado o resto da divisão de X por 1000000007 ($10^9 + 7$).

Restrições

- $1 \leq N \leq 10^6$
- $0 \leq M \leq 10^6$
- $1 \leq a_i \leq N$, para todo $i = 1, 2, \ldots, M$.
- $1 \leq b_i \leq N$, para todo $i = 1, 2, \ldots, M$.

Exemplos

Entrada	**Saída**
`2 0`	`1`

Entrada	**Saída**
`4 0`	`4`

Entrada	**Saída**

`4 2`
`1 3`
`2 2`
	`1`

Entrada	**Saída**
`3 0`	`2`

Entrada	**Saída**

`2 2`
`1 2`
`2 2`
	`0`

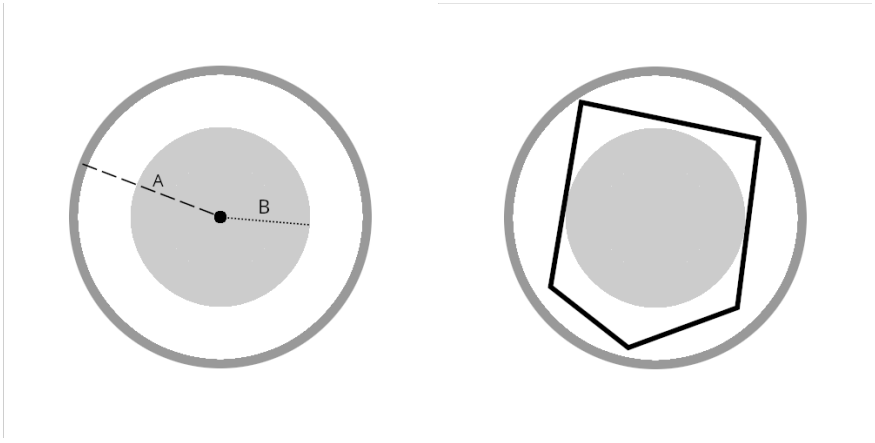
Problema I. Correndo em Círculos

Nome do arquivo fonte: `correndo.c`, `correndo.cpp`, *ou* `correndo.java`

A Associação de Competições Maratonísticas (ACM) entrou em contato com a organização da Maratona Mineira de Programação para registrar uma reclamação: por ter "Maratona" no nome, a competição estava recebendo muitas inscrições de corredores profissionais internacionais, em busca de pontos no ranking mundial de corridas. Porém, no dia anterior à competição, ao descobrirem que balões da Maratona Mineira não contam pontos no ranking da ACM, esses corredores se frustraram, retiraram sua inscrição e voltaram para casa. Para evitar esse mal entendido sem desperdiçar o potencial que a Maratona Mineira tem para cobrir outras modalidades, a ACM e a organização da Mineira chegaram a um acordo: a próxima edição da Mineira contará com uma corrida de 42km que somará pontos no ranking mundial da ACM!

Contudo, há um problema. Montar um bom circuito para uma Maratona é muito custoso, e a data da Mineira poderá colidir com outros eventos na cidade. Sendo assim, a organização decidiu realizar a corrida em uma praça circular. Afinal, a distância que pode ser percorrida dando voltas na praça é limitada apenas pela preparação do corredor!

Porém, pelas regras da ACM para minimizar lesões em corridas de longa distância, o circuito de corrida deve ser composto unicamente por segmentos de reta. As praças sendo consideradas pela organização possuem uma área interna, também circular, onde há um chafariz, e não é possível que o circuito passe por aí. O chafariz e a praça compartilham o mesmo centro. A praça tem raio A, e a área interna do chafariz tem raio B. Um circuito válido é um polígono que passa apenas pela área interna da praça externa ao chafariz, e que contém a área do chafariz.



Exemplo de praça com os raios A e B indicados, e um circuito possível composto de 5 segmentos de reta.

A organização da Maratona Mineira está interessada em saber qual é o menor número de segmentos de reta necessários para se formar um circuito válido em cada uma das praças que possivelmente serão escolhidas. Você pode ajudar?

Entrada

A entrada contém apenas uma linha com dois inteiros A e B, separados por espaço, representando o raio da praça e da região do chafariz, respectivamente.

Saída

Escreva na saída uma linha contendo um único inteiro N, o menor número de segmentos de reta necessários para formar um circuito válido na praça dada.

Restrições

- $1 \leq B < A \leq 10^9$

Exemplos

Entrada	**Saída**
`10 1`	`3`

Entrada	**Saída**
`10 9`	`7`

Problema J. Wasserman, Rei das Águas

Nome do arquivo fonte: `wasserman.c`, `wasserman.cpp`, *ou* `wasserman.java`

Em sua última visita a Furnas, Wasserman o Rei das Águas, ficou bastante confuso com o dialeto mineirês que os peixes falavam. Bagri, o peixe poliglota, foi convocado para ajudar Wasserman a entender esse novo idioma. Bagri passou muitos anos estudando o dialeto, e conhecia todas as palavras e frases mais comuns do Peixe-mineirês de Furnas. Fez então uma lista com todas elas e a deu à Wasserman. Aqui podemos ver um trecho desta lista:

$$\begin{aligned} \textit{blohblo} &: \text{aos competidores} \\ \textit{bloh} &: \text{ajuda} \\ \textit{blobla} &: \text{aos artistas da} \\ \textit{ubl} &: \text{Peça} \\ \textit{blaubl} &: \text{da Maratona Mineira} \end{aligned}$$

Então Bagri advertiu: - Sempre espere um peixe terminar sua frase, pois nem sempre uma palavra é o que parece! No peixe-mineirês quando há mais do que uma possível tradução, o correto é aquela em que as primeiras palavras são as maiores possíveis!

Wasserman, confuso, perguntou: - Quem poderá me ajudar a traduzir tal idioma?

Bagri, em peixe-mineirês, respondeu: - ublblohbloblaubl

Wasserman pensou, pensou, pensou até que disse: - A tradução para isso é *Peça ajuda aos artistas da Peça* ? Que Peça?

Bagri lamentando-se, disse: - Você está ficando velho Wasserman, não está entendendo muita coisa. Você traduziu como:

$$\textit{ubl} + \textit{bloh} + \textit{blobla} + \textit{ubl} : \text{Peça} + \text{ajuda} + \text{aos artistas da} + \text{Peça}$$

- Porém o correto é:

$$\textit{ubl} + \textit{blohblo} + \textit{blaubl} : \text{Peça} + \text{aos competidores} + \text{da Maratona Mineira}$$

- Pois em ambas traduções a primeira palavra tem o mesmo tamanho, porém na segunda tradução, a segunda palavra *blohblo* é maior do que a segunda palavra da primeira tradução *bloh*. Logo o correto é *Peça aos competidores da Maratona Mineira*!!!

Wasserman então pediu para que você faça um programa que traduza o peixe-mineirês.

Entrada

A primeira linha da entrada contém o inteiro N, representando o número de palavras de peixe-mineirês consideradas. As $2N$ linhas seguintes contém as palavras e suas traduções. A $(2i-1)$-ésima dessas linhas contém a i-ésima palavra em peixe-mineirês n_i, enquanto a $2i$-ésima delas

contém a tradução t_i da palavra n_i. Uma tradução é uma sequência de uma ou mais palavras separadas por espaço. Em seguida, a entrada contém mais uma linha apenas com letras (sem espaços) representando a sentença S a ser traduzida.

Saída

A saída deve conter uma única linha com a tradução de S. A tradução deve ser uma sequência de palavras separadas por espaço.

Restrições

- $1 \leq N \leq 5 \times 10^4$.
- $1 \leq |n_i| \leq 50$ para todo $i = 1, 2, \ldots, n$.
- $1 \leq |t_i| \leq 100$ para todo $i = 1, 2, \ldots, n$.
- $1 \leq |S| \leq 10^6$.
- Todas as palavras n_i são distintas.
- S contem apenas letras e possuí uma tradução válida.
- As palavras na entrada são formadas apenas por letras a-z ou A-Z.

Entrada	**Saída**
5 tiil WA qwIZh Cuidado abaka levar abakatiil levar TL pokkrjah para nao qwIZhpokkrjahabakatiil	Cuidado para nao levar TL

Entrada	**Saída**
5 dich te gewarnt avisei habe tenho Ich Eu Ichhabe Eu Ichhabedichgewarnt	Eu te avisei

Problema K. Cobra de Dominós

Nome do arquivo fonte: `cobra.c`, `cobra.cpp`, *ou* `cobra.java`

O menino Araújo, filho de biólogos, sempre foi fascinado pelas espécies de cobras que eram criadas no cativeiro da casa. Hoje pela manhã ele ganhou um dominó no bingo da escola e, já em casa, formulou o jogo "A Maior Cobra Legal de Dominós".

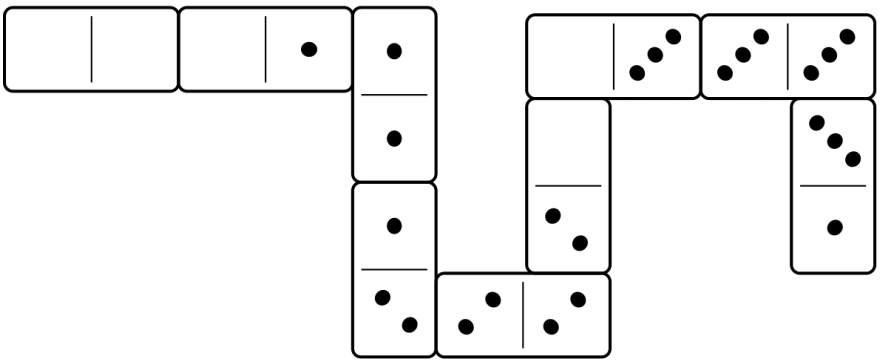
Uma peça de dominó é um par não ordenado de inteiros (i, j) tal que $0 \leq i, j \leq N$. Um jogo completo de dominó com N valores é composto por todas as peças de dominós possíveis que tem valores entre 0 e N. Uma cobra de dominó é uma sequência de peças $(i_1, j_1)(i_2, j_2) \ldots (i_m, j_m)$ tal que $j_k = i_{k+1}$ para todo k válido. O objetivo do jogo de nosso jovem protagonista é bem simples: formar a maior cobra de dominós possível usando peças de um jogo completo. Por exemplo, se seu jogo de dominó tem peças com valores de 0 até 3, uma possível maior cobra de se formar teria 9 peças com a seguinte configuração:

$$(0,0)(0,1)(1,1)(1,2)(2,2)(2,0)(0,3)(3,3)(3,1)$$

Observe que a peça 2-3 (ou 3-2, a peça é a mesma) não foi possível de ser usada. Por outro lado, se o valor de uma lado de suas peças vai até 4, uma possível maior cobra de se formar teria 15 peças com a seguinte configuração:

$$(0,0)(0,1)(1,1)(1,2)(2,2)(2,0)(0,3)(3,3)(3,1)(1,4)(4,4)(4,2)(2,3)(3,4)(4,0)$$

Curioso que é, o menino Araújo solicitou a ajuda dos programadores da Maratona Mineira para criar um programa com a seguinte missão: dado o valor máximo que pode aparecer em uma peça de dominó, informe qual é o tamanho da maior cobra legal de dominós.



Entrada

A entrada consiste de uma única linha contendo único inteiro N, o valor máximo que pode existir em uma peça de dominó.

Saída

A saída deve conter um único inteiro, que representa o tamanho da maior cobra de dominó que pode ser formada.

Restrições

- $0 \leq N \leq 5000$

Exemplos

Entrada	**Saída**
0	1

Entrada	**Saída**
3	9

Entrada	**Saída**
4	15

Problema L. História de Pescador

Nome do arquivo fonte: `pescador.c`, `pescador.cpp`, *ou* `pescador.java`

Senhor Figueiredo é um pescador famoso nos botecos da cidade de Lavras por causa das histórias duvidosas que ele conta sobre as aventuras dele no Rio Grande. Certa vez o jovem Ceconelli, vizinho do senhor Figueiredo, resolveu investigar a veracidade dos causos. Depois de muita insistência de Ceconelli, o senhor Figueiredo finalmente deixou que o jovem o acompanhasse, desde que carregasse os mantimentos e sua vara de pesca. Durante a pesca, Ceconelli acabou descobrindo o segredo do senhor Figueiredo sobre as histórias do pescador; o senhor Figueiredo multiplica a quantia fisgada por um fator fixo! Dados o fator multiplicativo usado pelo senhor Figueiredo e a quantidade real de peixes pescados, qual será a quantidade de peixes que o velho pescador irá contar para seus amigos no bar?

Entrada

A entrada contem uma única linha com os inteiros N e K, separados por espaço, que representam, respectivamente, a quantidade de peixes pescados e o fator usado pelo senhor Figueiredo nas histórias.

Saída

A saída contem um único inteiro M, a quantidade de peixes que o senhor Figueiredo dirá que pescou.

Restrições

- $1 \leq N, K \leq 1000$

Exemplos

Entrada	**Saída**
`10 5`	`50`