Seletiva para Maratona de Programação de 2014

Instituto de Matemática e Estatística

Universidade de São Paulo

Caderno de Problemas

Patrocínio:

Departamento de Ciência da Computação IME-USP

Sábado, 16 de agosto de 2014.

Instruções

- A competição tem duração de 5 horas;
- Faltando 1 hora para o término da competição, o placar será congelado, ou seja, o placar não será mais atualizado;
- Faltando 15 minutos para o término da competição, os times não receberão mais a resposta de suas submissões.

- A entrada de cada problema deve ser lida da entrada padrão (teclado);
- A saída de cada problema deve ser escrita na saída padrão (tela);
- Siga o formato apresentado na descrição da saída, caso contrário não é garantido que seu código será aceito;
- Na saída, toda linha deve terminar com o caracter '`\n`';
- O nome do arquivo de códigos em Java deve ser **exatamente** como indicado abaixo do nome de cada problema. Para C/C++ é recomendado usar o nome indicado;
- Para códigos em Java, o nome da classe principal deve ser **igual** ao nome do arquivo.

Problema A: Torneio de Yusuf II

Arquivo: `yusuf.[c|cpp|java]`

Se fazes tanta questão de ganhar, então joga sozinho.

Marrakech é uma das cidadelas do mundo árabe. A cidade foi fortificada nos anos 1122-1123 por Ali ibn Yusuf, filho de um dos primeiros dirigentes da cidade. Um dos grandes chefes da cidade foi o Califa Abu Yaqub Yusuf 'al-Mustansir' também conhecido como Yusuf II. O califa viveu apenas 21 anos (1203-1224) e reinou desde os seus 10 anos. Conhecido como um amante das artes e esportes, foi um grande incentivador da construção de palácios e museus. Idealizou uma grande competição de xadrez, uma de suas paixões, disputada por toda a corte.

O esquema imaginado por Yusuf II para determinar a classificação do campeonato de xadrez era muito interessante. Todos os N competidores jogavam contra todos. Um jogador K é chamado de ملك (malik) se para todo jogador J diferente de K, ou K vence J ou K vence algum jogador J' que vence J. A classificação do campeonato é uma ordenação dos jogadores $j_1, j_2, \ldots, j_N$ tal que cada jogador j_i, $1 \leq i < N$, vence j_{i+1} e é ملك se considerarmos os jogadores que aparecem depois dele na ordenação, isto é, ignorando os anteriores.

Sua tarefa será, dados os resultados dos jogos de um torneio de xadrez em Marrakech, determinar se é possível obter uma classificação de Yusuf II para aquele campeonato.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém um inteiro N correspondendo ao número de jogadores do torneio. Os jogadores são numerados de 1 a N. Para cada $i = 1, \ldots, N$, a $(i+1)$-ésima linha contém um inteiro $d_i \geq 0$ seguido pela identificação dos d_i jogadores vencidos pelo jogador i.

Saída

Para cada instância imprima uma linha com os jogadores ordenados de acordo com a classificação de Yusuf II. Caso exista mais de uma classificação possível, qualquer uma será aceita. Caso não exista classificação de acordo com o esquema de Yusuf II, imprima `inclassificavel`.

Restrições

- $2 \leq N \leq 1.000$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
3 1 2 1 3 1 1 3 2 2 3 0 1 2	1 2 3 1 3 2

Problema B: Desmascarando o empregado do sultão

Arquivo: `empregado.[c|cpp|java]`

Com a mentira se consegue o almoço, mas não o jantar.

Um sultão estava desconfiado de seu empregado. Ele dizia que trabalhava incessantemente, sem parar, para cumprir suas tarefas de N tipos diferentes. O sultão quer saber exatamente quanto tempo leva cada uma das tarefas, para poder avaliar melhor se o empregado é preguiçoso ou se realmente está sobrecarregado.

Para tentar desmascarar o empregado ele passou a solicitar relatórios de suas atividades. O empregado entregou N relatórios distintos, a mesma quantidade que o total de tarefas, o que deixou o sultão ainda mais desconfiado. Este empregado trabalha em jornadas de P horas e cada tarefa demora entre uma e P horas para completar. Todas as tarefas demoram uma quantidade inteira de horas para se completar. As jornadas de trabalho acontecem nas P primeiras horas do dia.

Cada relatório consistia da hora em que o empregado começou e a hora em que terminou de trabalhar. Se ele diz que começou a trabalhar às 0 horas e terminou as 3 horas, isso significa que ele começou no início daquela hora e terminou no final desta, totalizando 4 horas no período.

O empregado não anotou o dia em que começou a trabalhar e o dia em que terminou. Os horários do relatório nem sempre referem-se ao mesmo dia. Nesse caso o empregado afirma que parou de trabalhar no final do expediente e reinicionou a tarefa ao início do próximo dia. No exemplo anterior o empregado poderia ter trabalhado 4 horas, $P+4$ horas, $2P+4$ horas, etc. Com isso relatórios que indiquem que o empregado começou a trabalhar às 3 horas e terminou às 2 são perfeitamente válidos.

Além dessas informações, cada relatório contém quantas tarefas de cada tipo foram completadas. Durante esse período, o empregado afirma ter trabalhado sem parar.

Sua tarefa é, dadas as informações dos relatórios, determinar qual a duração, em horas, de cada tarefa, caso isso seja possível.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém dois inteiros, N e P. As próximas N linhas contêm $N+2$ inteiros cada uma. A i-ésima linha, dessas N linhas, corresponde ao i-ésimo relatório e consiste de S_i, T_i, $A_{i,1}$, ..., $A_{i,N}$. Onde S_i e T_i correspondem, respectivamente, à hora em que o empregado começou e parou de trabalhar. Cada $A_{i,j}$ é a quantidade de vezes que a tarefa j foi realizada no período do relatório i.

Saída

Para cada instância imprima uma única linhas na saída, que consiste de:

- "-1" caso você possa afirmar com certeza que o empregado tenha mentido em algum relatório;
- "-2" caso você não possa afirmar que o empregado esteja mentindo, mas também não possa encontrar as durações de cada máquina de maneira única;
- N inteiros, separados por espaço, que representem a duração de cada tarefa se essas durações puderem ser determinadas de forma única.

Restrições

- $1 \leq N \leq 100$
- $2 \leq P \leq 24$
- $0 \leq S_i, T_i < P$
- $0 \leq A_{i,j} \leq 10^9$
- $\sum_j A_{i,j} > 0$
- P é um número primo

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
3 23 1 5 5 0 0 3 6 0 0 1 0 22 3 2 2 2 7 1 2 1 0 2 5 1 0 2 13 1 3 1 1 1 6 2 2	1 6 4 -1 -2

Problema C: O verdadeiro valor dos tapetes árabes

Arquivo: `tapete.[c|cpp|java]`

Sábio é quem estende seu manto como se fosse tapete, e tolo é quem pisa.

Os tapetes árabes são muito conhecidos. Sua qualidade é reconhecida em todo o mundo, e as características de um bom tapete são apreciadas por todos. Avaliar os tapetes é uma tarefa muito difícil, e os especialistas analisam suas características minuciosamente para estabelecer um preço adequado. Os tapetes são formados por pontos, onde os fios são amarrados. Apesar de ser muito difícil para uma pessoa comum, os especialistas são capazes de dizer a direção em que o fio foi amarrado entre dois pontos. Estes fios formam circuitos nos nós e padrões complicados têm centenas ou mesmo milhares de circuitos e são muito intrincados. Circuitos grandes (em que a quantidade de fio dividida pelo número de nós é muito grande) desvalorizam o tapete, pois o torna menos resistente. Circuitos pequenos são valorizados, e o avaliador sempre busca encontrar o menor circuito existente no tapete, pois este é um indicador do valor do tapete. Sua tarefa neste exercício é ler os dados de um tapete com N nós e M fios (ligações entre estes nós em que a direção em que foi feita é determinada) e determinar o valor do menor circuito do tapete, ou seja, o circuito em que a razão entre a quantidade de fio dividida pelo número de nós é mínimo.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém dois inteiros, N e M, correspondendo aos números de nós e ligações, respectivamente. Os nós são numerados de 1 a N. Seguem M linhas, cada uma com três inteiros u, v e c descrevendo uma ligação do nó u para o nó v usando c cm de fio.

Saída

Para cada instância, imprima em uma única linha o valor mínimo de um circuito do tapete, onde esse valor é a razão entre a quantidade de fio dividida pelo número de nós no circuito. O valor deve ser impresso com 3 casas decimais. Imprima -1, caso não exista circuito no tapete.

Restrições

- $3 \leq N \leq 1.000$
- $N \leq M \leq N \times N - 1$
- $0 \leq c \leq 1.000$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
3 3 2 1 4 3 1 2 3 2 1 4 5 1 2 40 2 3 20 3 4 20 4 1 30 4 2 50	-1 27.500

Problema D: Picos do Átlas

Arquivo: `picos.[c|cpp|java]`

Muro baixo, o povo pula.

O Marrocos é cortado pelas montanhas Átlas, cujo pico mais alto é Toubkal, com 4.165 metros. Estas montanhas deram origem a diversos mitos e histórias durante toda a antiguidade, como, por exemplo, nos 12 trabalhos de Hércules. Próximo à cidade de Marrakech fica o que se chama de "alto Átlas", a parte mais alta destas montanhas.

O estudo das altitudes dos diversos picos tem sido feito há séculos. Antigos documentos berberes documentam o registro de diferentes altitudes dos vários pontos das montanhas Átlas desde o século XVI. O documento é um mapa da região dividido em quadrantes. Em cada quadrante está anotada a altura média daquele ponto. Sabemos que um ponto é um pico se a altura daquele quadrante é maior que de seus vizinhos (um quadrante tem até 8 vizinhos).

Sua tarefa neste exercício é ler esse mapa e identificar os picos existentes na região documentada.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância corresponde ao mapa de uma região e é representado por uma matriz $N \times M$. A primeira linha de cada instância contém os inteiros N e M. Para $i = 1, 2, \ldots, N$, a $(i+1)$-ésima linha corresponte a i-ésima linha da matriz e contém M inteiros separados por um espaço.

Saída

Para cada instância imprima as coordenadas dos picos do mapa correspondente, uma por linha, ordenado primeiro pelas linhas e, em caso de empate, pelas colunas. Caso não exista nenhum pico, imprima `-1`. Imprima uma linha em branco no final da saída de **toda** instância.

Restrições

- $1 \leq N, M \leq 1.000$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**

`3 3`
`2 1 1`
`1 1 6`
`4 1 0`
`3 3`
`1 1 1`
`1 3 1`
`1 1 3`
	`1 1`
	`2 3`
	`3 1`
	
	`-1`

Problema E: O gato do zelador do armazém

Arquivo: `gato.[c|cpp|java]`

Quando disseram ao gato que o seu excremento era útil ele começou a enterrá-lo.

Sokoban é um jogo muito conhecido. O que poucas pessoas sabem é que o jogo foi inventado pelos bérberes, povos árabes que habitavam o norte da África, mais especificamente onde hoje existe o Marrocos. Para estes povos os gatos são considerados um animal que traz mau agouro, principalmente por conta de seu comportamento egoísta (como atesta o provérbio destacado acima).

Nessa versão original do sokoban o personagem é um gato que fica empurrando os blocos de terra ao seu redor. Há um tabuleiro que consiste de N linhas e M colunas. Você controla o gato que pode se movimentar em qualquer uma das 4 casas adjacentes, desde que essa esteja livre, ou pode empurrar um bloco em qualquer uma dessas 4 direções.

Como na versão mais conhecida, seu objetivo é empurrar o bloco até uma posição final especificada. Note que você pode apenas empurrar o bloco e nunca puxar.

A versão bérbere do jogo tem uma particularidade. Há algumas casas especiais que são portas. Sempre que o gato ocupa uma posição com porta pela primeira vez, a porta é aberta. O bloco não pode ser empurrado para a posição de uma porta se ela estiver fechada. Após aberta, uma porta se comporta como uma posição livre.

Você quer empurrar o bloco para a posição final abrindo o menor número de portas possível.

Considere toda posição fora do tabuleiro como uma parede.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância inicia com uma linha contendo 2 inteiros separados por um espaço, N e M. Seguem-se N linhas, cada uma com M caracteres pertencentes ao conjunto $\{*, ., j, b, x, \#\}$, onde:

- '*' representa uma parede;
- '.' representa uma posição vazia;
- 'j' representa a posição inicial do gato;
- 'b' representa a posição inicial do bloco;
- 'x' representa a posição final desejada para o bloco;
- '#' representa uma porta;

Saída

Para cada instância imprima uma única linha. Caso seja possível empurrar o bloco para a sua posição final essa linha deverá consistir de 2 inteiros, que são o número mínimo de portas que precisam ser abertas para realizar tal ação e o número mínimo de movimentos que são necessários com esse número de portas abertas. Imprima '-1' caso não seja possível empurrar o bloco para a sua posição final.

Restrições

- $1 \leq N, M \leq 25$
- Cada instância contêm exatamente um caractere 'j'.
- Cada instância contêm exatamente um caractere 'b'.
- Cada instância contêm exatamente um caractere 'x'.
- Cada instância contêm no máximo 5 caracteres '#'.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
1 4 jb.x 1 4 bj#x 3 5 .j..x .b*#* *####	0 2 -1 4 12

Problema F: Partição do rebanho

Arquivo: `rebanho.[c|cpp|java]`

Visita sem presentes é melhor do que a que te traz um carneiro.

A culinária marroquina é muito famosa por suas deliciosas receitas que envolvem vários tipos de carnes assadas, mas especialmente carneiros, que são criados na região desde o século VIII. Uma curiosa tradição bérbere envolve a partilha da criação de um pastor no momento de sua morte. Independentemente do número de filhos que ele tenha, apenas o primogênito e o filho mais novo têm direito à herança. Os demais filhos não ganham nada. Então, todos os animais são pesados, e os pesos (arredondados para o inteiro mais próximo) são considerados. O rebanho é então dividido em duas partes, de forma que, em cada uma os animais têm pesos semelhantes. Mais especificamente, o rebanho é particionado em duas partes, A e B, de tal forma que,

$$S(A,B) = \sum_{a_1,a_2 \in A} |\ \text{peso}(a_1) - \text{peso}(a_2)\ | + \sum_{b_1,b_2 \in B} |\ \text{peso}(b_1) - \text{peso}(b_2)\ |$$

seja mínima. Então, o primogênito fica com a parte do rebanho de peso maior, e o último filho, com a parte de peso menor. Não parece muito justo, mas é a tradição por lá.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém um inteiro N indicando a quantidade de carneiros no rebanho. A linha seguinte contém N inteiros separados por um espaço, correspondendo aos pesos dos carneiros.

Saída

Para cada instância, imprima em uma única linha o valor mínimo de $S(A,B)$.

Restrições

- $2 \leq N \leq 1.000$
- $0 \leq \text{peso}(\cdot) \leq 100$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
4 1 4 4 1 4 1 2 3 4	0 2

Problema G: Hiperprimos

Arquivo: `hiperprimos.[c|cpp|java]`

Ele começou a multiplicar os quintos pelos sextos...

Várias descobertas matemáticas da idade média são devidas a matemáticos árabes famosos como al-Khwārizmī [1], Omar Khayyám, e Sharaf al-Dīn al-Ṭūsī entre outros. Um dos resultados pouco conhecido é sobre os números hiperprimos. Dizemos que um número é hiperprimo se ele tiver um número primo de divisores. Assim, por exemplo, 25 é hiperprimo, pois tem 3 divisores. Já 42 não é hiperprimo, pois tem 8 divisores.

Dado um inteiro N, determine o número de hiperprimos no intervalo $[2, N]$.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância consiste de uma única linha contendo um único inteiro, N.

Saída

Para cada instância, imprima uma linha com a quantidade de hiperprimos no intervalo $[2, N]$.

Restrições

- $2 \leq N \leq 2 \times 10^6$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 4	1 3

[1] Uma das transliterações de seu nome para o latim foi Algoritmi e deu origem às palavras algoritmo e algarismo. Além disso, álgebra deriva da palavra árabe *al-jabr*, usada em um de seus trabalhos.

Problema H: Canais de qanat

Arquivo: `canais.[c|cpp|java]`

– Teu moinho gira para a direita ou para a esquerda?
– Sei lá, o importante é que ele me dá farinha!

A medina de Marrakech é formada pela cidade fortificada, patrimônio universal da Unesco desde 1995. O início de sua construção remonta à fundação da cidade no século XI e inclui vários monumentos impressionantes, como a mesquita de Koutoubia, madraçal de Ben Youssef, e o Palácio Bahia. Várias histórias cercam os monumentos que formam a medina. A mais interessante diz respeito os jardins Ménara. O parque tem hortas e lagos artificiais construídos na época do sultão Abd-el-Rhaman, que era um apaixonado por desafios matemáticos. Um dos mais brilhantes é o dos conjuntos de canais de qanat (قناة). Cada conjunto é formado por um canal fechado e um canal aberto. O canal fechado tem o formato de um polígono e o canal aberto consiste de uma sequência de arestas formando um caminho. O desafio é determinar se é possível transformar o canal fechado no canal aberto através de operações mentais sobre o canal fechado, como de remoção de vértices e arestas, translações e rotações (no plano).

Os canais são dados através das coordenadas dos seus vértices e a ordem na qual os vértices são dados indica o sentido do fluxo de água. Consideramos que é possível transformar o canal fechado no aberto se após a aplicação das operações, o canal resultante tem as mesmas coordenadas e a água flui no mesmo sentido.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém dois inteiros N_f e N_a, correspondendo ao número de vértices dos canais fechado e aberto, respectivamente. A linha seguinte contém N_f pares de inteiros (x_i, y_i), cada par representando a coordenada de um vértice do canal fechado. No canal fechado a água sempre flui no sentido anti-horário e os vértices são números de 1 a N_f. A terceira e última linha contém N_a pares de inteiros (x_i, y_i) correspondendo aos vértices do canal aberto.

Saída

Para cada instância imprima `-1`, se não é possível transformar o canal fechado no aberto, ou o menor índice do vértice do canal fechado que coincide com primeiro vértice do canal aberto após a transformação.

Restrições

- $3 \leq N_f \leq 20.000$
- $2 \leq N_a \leq 5.000$
- $-10.000 \leq x, y \leq 10.000$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 2 0 0 4 3 0 6 -4 3 2 -1 2 4 4 3 0 0 4 3 0 6 -4 3 5 0 5 5 4 5	1 -1

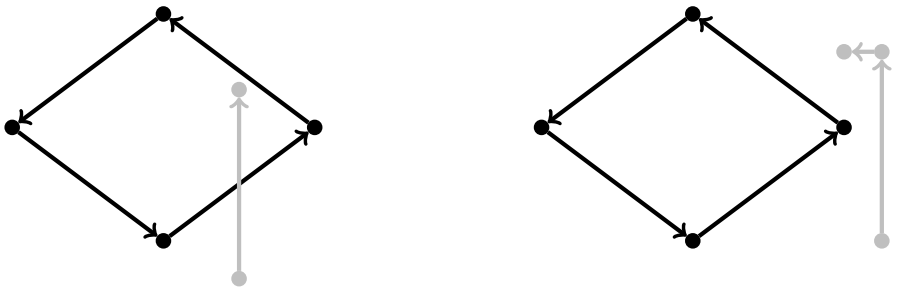

Figura 1: Ilustração primeiro (esq.) e segundo (dir.) exemplos de entrada.

Problema I: *Pair-voting* no conselho de Gueliz

Arquivo: `pairvoting.[c|cpp|java]`

Antes de examinar a casa (para comprar), examina os vizinhos.

O bairro de Gueliz em Marrakech é hoje conhecido por ser a parte moderna da cidade, com diversas opções turísticas, restaurantes e bares. Poucos conhecem a formação do bairro, ainda no século XVI. Originalmente o bairro, também conhecido como "Cidade Nova" foi se formando fora da fortaleza (Medina, cidade antiga). O primeiro novo morador ganhou uma autorização oficial da prefeitura para construir sua casa, e ficou responsável por dar novas autorizações. Quando uma rua foi aberta, um habitante do fim da rua (até a primeira esquina formada) foi designado representante da rua juntamente com o primeiro morador. E assim ocorreu para todas as ruas da cidade: os moradores das esquinas eram representantes das ruas que se encontravam naquela esquina, de forma que cada trecho de rua sem esquinas tem exatamente dois representantes. Há em Gueliz uma lenda que impede a formação de quarteirões (conjunto de casas cercadas por ruas). Os antigos contam que uma vez formaram um quarteirão no bairro, e quando uma pessoa má morreu seu espírito ficou ali preso para sempre, assombrando as pessoas que ali moravam. Desde então nunca mais se formaram quarteirões no bairro.

O conselho do bairro de Gueliz é formado pelo primeiro morador e os representantes de cada rua. Estes representantes formam comitês para analisar as diversas questões. Nos comitês os conselheiros são agrupados em pares, e todos os conselheiros devem participar de exatamente um par. Cada par tem um único voto e a moção é aprovada quando atinge maioria dos votos. Cada par deve ser formado por conselheiros representantes de uma mesma rua (ou pelo primeiro morador e um representante de uma rua que faz esquina com ele). Claramente, quando o número de conselheiros é ímpar não é possível encontrar uma composição dos comitês, em pares, com a participação de todos os conselheiros. Quando isso ocorria, todos concordavam que o primeiro morador podia ter um voto sozinho, e os demais deveriam ser divididos em pares.

Entretanto, com o passar do tempo houve ocasiões em que não foi possível montar um comitê, o que sempre foi motivo de desconfiança entre os moradores de Gueliz. Sua tarefa neste exercício é dado N o número de representantes de rua (o representante 1 é o primeiro morador) decidir se é possível formar um comitê de pares de conselheiros conforme descrito acima.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém um inteiro N. As próximas $N-1$ linhas contêm 2 inteiros cada uma. A i-ésima linha, dessas $N-1$ linhas, contém os representantes x e y, $x \neq y$, $(1 \leq x, y \leq N)$ de um trecho de rua sem esquinas.

Saída

Para cada instância, imprima na primeira linha '`Y`' se é possível formar um comitê de pares de conselheiros ou '`N`', caso contrário. Caso seja possível formar um comitê, imprima uma lista de pares de conselheiros, um par por linha. Um par de conselheiros consiste de dois inteiros x_i e y_i, separados por um espaço, de forma que $x_i < y_i$. Além disso, a lista de pares de conselheiros deve estar ordenada de forma crescente por x_i. Caso exista mais de uma forma de montar um comitê, imprima a lexicograficamente menor. Note que, quando o representante 1 tem voto sozinho, ele não pertence a nenhum par.

Restrições

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

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
8 1 2 1 5 2 4 1 3 5 7 5 8 3 6 10 1 2 1 3 1 4 1 7 3 5 4 6 7 9 7 8 9 10	N Y 1 2 3 5 4 6 7 8 9 10

Problema J: Montando sua própria cáfila

Arquivo: `cafila.[c|cpp|java]`

O camelo que quer ver sua corcova, torce o pescoço.

Camelos foram domesticados e são utilizados pelos povos beduínos há vários milênios. Há dois tipos principais, os camelos bactrianos, com 2 corcovas, que são mais raros, e os dromedários, mais comuns, com apenas uma corcova. Para simplificar, os camelos bactrianos são chamados apenas de camelos.

Tanto camelos quanto dromedários têm diferentes características interessantes: alguns são líderes natos, outros têm mais força, outros suportam caminhadas mais longas, etc. No entanto, cada animal possui apenas uma dessas características. Uma caravana precisa de animais com várias dessas características. No comércio desses animais são consideradas N_1 características diferentes para dromedários e N_2 características diferentes para camelos.

Os beduínos comercializam esses animais sempre em cáfilas formadas por grupos de três. Essa é uma tradição bérbere que remonta ao século IX, quando os comerciantes de camelos e dromedários só podiam vender seus animais dessa forma. Os compradores não podem escolher os animais. Os lotes de 3 animais são preparados pelo vendedor e o comprador pode apenas dizer o número do lote que deseja comprar, sem mais informações a respeito. Os comerciantes são conhecidos por sua extrema honestidade, e cada lote é formado de forma aleatória. Para cada animal do lote, o comerciante sorteia com probabilidade p_1 se será um dromedário e com probabilidade $p_2 = 1 - p_1$ se será um camelo, $p_1 \geq p_2$. Escolhido o tipo i de animal, é escolhida uma característica dentre as N_i, com igual probabilidade, e um animal com essa característica é incluído no lote.

Todos os sorteios do comerciante são independentes, podendo um lote conter dois dromedários fortes e um camelo líder, por exemplo, ou mesmo três camelos bons de caminhada. Entretanto, para atestar sua honestidade, sempre que o comerciante produz um lote que tem apenas dromedários ele faz um novo sorteio.

Dentre as N_1 características para dromedários, há M_1 que são desejáveis para o comprador. E dentre as N_2 características para camelos, há M_2 que são desejáveis para o comprador. Qual o número esperado de lotes que um comprador deve adquirir para montar uma cáfila que tenha dromedários e camelos com todas as M_1 e M_2 características desejadas?

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância consiste de 3 linhas, cada uma contendo 2 inteiros. Na primeira linha temos N_1 e N_2. A segunda linha consiste de M_1 e M_2, enquanto que na terceira linha temos W_1 e W_2, que dão as probabilidades de cada categoria através da relação:

$$p_i = \frac{W_i}{W_1 + W_2}$$

Saída

Para cada instância imprima uma linha contendo um número com 3 casas decimais, que é o número esperado de lotes que o comprador precisa comprar para completar sua cáfila.

Restrições

- $1 \leq N_i \leq 50$
- $0 \leq M_i \leq N_i$
- $1 \leq W_2 \leq W_1 \leq 100$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
1 1 0 0 1 1 1 7 1 0 1 1 5 5 3 2 11 6	0.000 1.167 7.677

Problema K: As dicas de Ali Babá

Arquivo: `alibaba.[c|cpp|java]`

Aperta-lhe a mão, mas confere os dedos depois.

As mil e um noites são uma coletânea de histórias árabes que remontam ao século IX. Algumas traduções para o ocidente foram feitas a partir do século XVII, e algumas destas histórias, como "Simbad, o marujo", "Aladim e a lâmpada mágica" e "Ali Babá e os quarenta ladrões" são hoje conhecidas por crianças de todo o mundo. Na história de Ali Babá os tesouros são guardados em uma gruta que se abre quando a expressão "Abre-te Sésamo" é usada. Na verdade, o tesouro estava escondido dentro de um cofre na parede da gruta, que se abria quando uma permutação dos inteiros de 1 a N era recitada. Nem todos os 40 ladrões tinham boa memória, assim Ali Babá era obrigado a manter nas paredes da gruta, dicas de como reconstruir a permutação, caso algum dos ladrões a esquecesse. Ele anotava uma sequência de inteiros $a_1, a_2, \ldots, a_k$ gerada a partir da permutação que abria o cofre após possíveis aplicações das seguintes operações: duplicação(i, j) e espelhamento(i, j), para $i \leq j$. A operação duplicação(i, j) cria uma cópia da subsequência a_i, a_{i+1}, $\ldots$, a_j e a insere entre a_j e a_{j+1}. A operação espelhamento(i, j) insere uma cópia invertida da subsequência de a_i até a_j (a_j, a_{j-1}, $\ldots$, a_i) entre a_j e a_{j+1}.

Por exemplo, dada a sequência $(a_1, a_2, a_3, a_4, a_5, a_6)$, a aplicação da operação espelhamento$(3, 5)$ gera a sequência $(a_1, a_2, a_3, a_4, a_5, a'_5, a'_4, a'_3, a_6)$.

Sua tarefa é reconstruir a permutação original.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém os inteiros K e N indicando, respectivamente, o tamanho da sequência escrita por Ali Babá e o maior inteiro da permutação original. A linha seguinte contém os K inteiros da sequência, separados por um espaço.

Saída

Para cada instância, imprima uma única linha com a permutação que originou a sequência da entrada, com um espaço separando inteiros consecutivos. Caso exista mais de uma permutação possível, qualquer uma delas será aceita.

Restrições

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

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**

`4 2`
`1 2 1 2`
`5 3`
`1 2 2 1 3`
	`1 2`
	`1 2 3`