IV Maratona Mineira de Programação

Caderno de Problemas — Competição Principal

Varginha, 13 de Junho de 2015

Instruções:

- Este caderno contém 12 problemas: as páginas estão numeradas de 1 a 14, não contando esta página de rosto. Verifique se o caderno está completo.
- Em todos os problemas, a entrada de seu programa deve ser lida da *entrada padrão*. A saída deve ser escrita na *saída padrão*.

Problema A. Quase Primos

Arquivo: `quaseprimos.c`, `quaseprimos.cpp` ou `quaseprimos.java`
Limite de tempo: 1 segundo

Um número inteiro positivo é primo se ele possui exatamente dois divisores: 1 e ele mesmo. Um número que possui exatamente 3 divisores é um "quase primo".

Dado um inteiro positivo, determine se ele é quase primo ou não.

Entrada

A entrada possui um único inteiro k.

Saída

Imprima uma linha contendo `S` se k é um quase primo, ou `N` caso contrário.

Limites

$1 \leq k \leq 1000$

Exemplos

Entrada	**Saída**
9	S

9 possui três divisores: 1, 3 e 9.

Entrada	**Saída**
31	N

31 é primo, e portanto possui apenas dois divisores.

Entrada	**Saída**
55	N

55 possui quatro divisores: 1, 5, 11 e 55.

Problema B. Metade da Sua Idade Mais Sete

Arquivo: `metademaissete.c`, `metademaissete.cpp` ou `metademaissete.java`
Limite de tempo: 1 segundo

Algumas pessoas na sociedade não consideram apropriados relacionamentos entre duas pessoas quando a diferença de idade entre elas é muito grande. Outras acham que isso é uma grande bobagem, e desde que ambas as pessoas estejam de acordo e que não haja impedimento legal, qualquer relacionamento é ok.

Não é claro, porém, como definir o que é uma diferença aceitável ou não. Alguns casos são bem simples: quase todo mundo consideraria um relacionamento entre uma pessoa de 77 anos e outra de 19 impróprio. Mas outros casos não são mais tão claros. Um relacionamento entre alguém de 25 e alguém de 35 é válido? Uma regra simples que existe desde pelo menos 1901 é que você não deve se relacionar com ninguém que tenha menos do que *metade da sua idade mais sete*[1], arredondando para baixo.

Por exemplo, se você tem 22 anos, então por essa regra seu limite inferior é 18 anos. Alguém de 31 anos pode se relacionar com pessoas de 22 ou mais — nesse caso, metade da idade mais sete dá 22,5, mas o valor é sempre arredondado para baixo. E assim por diante.

A regra, como não poderia deixar de ser, é bem controversa. Algumas pessoas acham ela restritiva demais. Outras já acham ela permissiva demais. Outras já acham que não faz sentido nenhum sequer ter uma regra. Nós não queremos aqui dizer que ela é certa nem errada, nem que você deve ligar para ela. Mas ela tem algumas propriedades interessantes.

Uma delas é que duas pessoas quaisquer sempre vão poder se relacionar, mesmo que tenham que esperar alguns anos antes do início. Por exemplo, suponha que haja duas pessoas, uma com 40 anos, e a outra com 20. Pela regra, esse relacionamento é impróprio, pois o limite inferior para alguém de 40 anos é 27. Mas se essas duas pessoas esperarem 13 anos, então elas poderão se relacionar sem problemas: terão, então, 53 e 33 anos, e o limite inferior para alguém de 53 anos é 33.

Dadas as idades de duas pessoas, calcule quantos anos elas terão que esperar antes de começar um relacionamento, de acordo com essa regra.

Entrada

A entrada possui apenas uma linha, contendo dois inteiros a e b, que representam as idades das duas pessoas, em anos.

Saída

Imprima uma linha contendo o número mínimo de anos que essas pessoas devem esperar antes de iniciar um relacionamento.

Observações

Considere que as duas pessoas fazem aniversário no mesmo dia, e esse dia é hoje.

Limites

$13 < a, b \leq 100$

Exemplos

Entrada	Saída
22 21	0

O limite inferior para alguém de 22 anos é $\frac{22}{2} + 7 = 18$, logo a relação já é apropriada de acordo com a regra, e os pombinhos não precisam esperar nem um ano.

Entrada	Saída
35 22	4

O limite inferior para alguém de 35 é 24. Após quatro anos, as pessoas terão, respectivamente, 39 e 26 anos, e o relacionamento se torna próprio de acordo com a regra.

[1] http://xkcd.com/314/

Problema C. Comprando Pregos

Arquivo: `pregos.c`, `pregos.cpp` ou `pregos.java`
Limite de tempo: 1 segundo

Na loja de pregos de Pedro, tudo é muito organizado. No entanto, Pedro é uma pessoa cheia de manias, e não organiza as coisas de maneira muito tradicional. Um exemplo disso é seu estoque de pregos, que fica guardado em uma fila de N caixinhas. Ordenadas da esquerda para a direita, cada caixinha contém exatamente a quantidade de pregos correspondente à sua posição. Isso quer dizer que a primeira caixa contém 1 prego, a segunda contém 2 pregos e assim por diante, até a enésima caixa, que contém N pregos.

Essa forma pouco usual de organizar o estoque ainda não é suficiente para satisfazer as manias de Pedro. Ele decidiu que não venderá pregos individuais, mas apenas caixinhas de pregos fechadas. Como consequência, Pedro agora precisa descobrir quais pedidos ele será capaz de atender. Infelizmente, ele não é muito bom em matemática e não sabe programar. Felizmente, você estava sem problemas divertidos para resolver e decidiu ajudá-lo!

Você deverá implementar um programa para, dada uma consulta do tipo (N, Q), determinar se Pedro consegue atender um pedido de exatamente Q pregos, sendo que o estoque de Pedro possui N caixas. Caso seja possível atender o pedido, o programa deverá encontrar o menor número de caixas possível que Pedro precisará vender.

Entrada

A entrada é formada por K consultas. Cada linha da entrada especifica uma consulta e contém dois inteiros, N e Q, respectivamente, o número de caixas do estoque de Pedro e a quantidade de pregos do pedido.

Saída

Para cada consulta, imprima uma linha com um inteiro que representa o menor número de caixas que Pedro precisa vender para satisfazer o pedido. Caso não seja possível atender o pedido, imprima "IMPOSSIVEL", sem aspas e sem acento.

Limites

$1 \leq K \leq 10^6$

$1 \leq N \leq 10^9$

$1 \leq Q \leq 10^{18}$

Exemplos

Entrada	**Saída**
4 4 30 3 4 20 15	IMPOSSIVEL 1 2

Problema D. Sangue

Arquivo: `sangue.c`, `sangue.cpp` ou `sangue.java`
Limite de tempo: 1 segundo

Leonardo sempre foi um garoto muito interessado em genética. Recentemente, na escola, aprendeu que o tipo sanguíneo de uma pessoa (A, B, AB e O) é definido unicamente através de genes recebidos diretamente pelos pais. Para determinar o tipo sanguíneo, cada pessoa recebe exatamente um gene do pai e um da mãe, que podem ser de 3 tipos: a, b, o.

Se uma pessoa receber, os genes *ao* ou *aa* ela terá tipo sanguíneo "A". Caso receba *bo* ou *bb* ela terá tipo sanguíneo "B". Por fim, se receber *oo* terá tipo sanguineo "O" e se receber *ab* terá tipo "AB". Note que a ordem de recebimento dos genes não importa. Com essas anotações Leonardo descobriu que é possivel descobrir ou, pelo menos, estimar o tipo sanguíneo dos filhos dados os tipos dos pais. Ele pediu para que voce determinasse a probabilidade do tipo sanguineo de uma pessoa dados os tipos sanguineos dos(as) avós.

Entrada

A entrada é composta por uma linha. A entrada possui 4 strings ("A", "B", "AB" ou "O") separadas por espaço. As primeiras duas strings se referem aos avós Maternos, as duas seguintes se referem aos avós paternos.

Saída

Imprima uma única linha na saída, contendo 4 números de ponto flutuante indicando, respectivamente, a probabilidade do indivíduo de ter tipo sanguíneo 'A", "B", "AB" ou "O". Cada número deverá conter apenas duas casas decimais.

Observações

- Todas as pessoas tem exatamente dois genes e eles possuem a mesma chance de serem transmitidos para os filhos.
- Ignore todos os antepassados dos avós. A chance de aparecemento dos genes dado o tipo sanguineo. Ou seja, se uma avó for do tipo A, por exemplo, ela tem garantidamente um gene "a" e tem 0.5 de probabilidade de ter outro gene "a" e 0.5 de probabilidade de ter um gene "o".
- Por questões de arredondamento, nem sempre as probabilidades vão somar 1.0.

Exemplos

Entrada	**Saída**
A B O O	0.38 0.38 0.00 0.25

Entrada	**Saída**
AB O O A	0.44 0.16 0.09 0.31

Entrada	**Saída**
A A A A	0.94 0.00 0.00 0.06

Problema E. Uno

Arquivo: `uno.c`, `uno.cpp` ou `uno.java`
Limite de tempo: 1 segundo

Uno é um viciante jogo de cartas cujo objetivo é descartar suas cartas o mais rápido possível, já que ganha quem fica com a mão vazia. Um objetivo secundário, e aí está a parte viciante do jogo, é usar cartas especiais contra seus adversários. Existem cartas que forçam o adversário a comprar mais cartas, perder a vez, e ainda inverter o sentido do jogo!

Como o jogo é bastante dinâmico, os iniciantes ficam perdidos. Não há coisa mais irritante para os jogadores experientes que ficar esperando o amigo jogar e ele perguntar "de quem é a vez agora?". Seu objetivo nesta questão é ajudar esses iniciantes perdidos, informando de quem é a vez de jogar a cada vez que alguém pergunta.

O jogo começa no sentido horário: jogador 1, jogador 2, jogador 3... até jogador N, em seguida, novamente, jogador 1, jogador 2, ... Uma carta '`A`' (ás) faz o próximo jogador perder a vez, passando a vez para o seguinte. Uma carta '`7`' faz o próximo jogador comprar cartas e não jogar (ou seja, perde a vez, passando a vez para o seguinte). Uma carta '`Q`' (dama) inverte o sentido do jogo: se seguia no sentido horário passa a seguir no sentido anti-horário, e vice-versa. As demais cartas não tem efeito sobre quem é o próximo jogador.

Considere o seguinte exemplo de sequência de jogadas em uma partida de Uno, para 5 jogadores. A primeira linha indica o jogador da vez e a segunda linha a carta que ele descartou, com as cartas especiais marcadas.

```
1 2 3 4 5 1 2 4 5 2 3 2 1 5 3 2 1 5 3 2 3 2 1 5 4 ...
2 5 6 4 2 9 A 8 A 4 Q 3 9 A 8 4 6 7 2 Q Q 6 4 3 8 ...
```

Quando o jogador 2 descarta o primeiro '`A`', o jogador seguinte (o 3) perde a vez. Quando o jogador 5 descarta seu primeiro '`A`', o jogador seguinte (o 1) perde a vez. O jogador 3 inverte o sentido do jogo com a carta '`Q`', que passa a seguir no sentido anti-horário. O jogador 5 descarta outro '`A`' fazendo o jogador seguinte (o 4) perder a vez. O jogador 5 descarta um '`7`' que faz o jogador seguinte (o 4) comprar carta e perder a vez. O jogador 2 descarta um '`Q`', invertendo o sentido do jogo, e o próximo (o 3) também descarta um '`Q`', invertendo o sentido novamente.

Entrada

A entrada tem duas linhas. A primeira contém um inteiro N, o número de jogadores da partida. A segunda contém uma sequência de caracteres separados por espaço em branco. Esses caracteres podem ser uma carta (`A 2 3 4 5 6 7 8 9 T J Q K`) ou uma pergunta (`?`). Apenas as cartas `A 7 Q` tem efeito no jogo, sendo que as duas primeiras tem o mesmo efeito. A entrada termina com um caractere `#`.

Saída

Para cada pergunta da entrada, ou seja, a cada `?`, escreva o número do jogador da vez.

Escreve um espaço em branco após o número, inclusive após o último.

Limites

$3 \leq N \leq 20$

Exemplos

Entrada
5 ? 2 5 6 4 2 9 A ? 8 A 4 Q ? 3 9 A 8 4 6 7 2 Q Q 6 4 3 8 ? #
Saída
1 4 2 3

Entrada
10 T K ? 7 ? 5 6 J 8 ? Q Q Q A ? 9 9 9 9 A ? A A A Q ? 3 4 #
Saída
3 5 9 6 10 5

Problema F. Maior palavra

Arquivo: `palavras.c`, `palavras.cpp` ou `palavras.java`
Limite de tempo: 1 segundo

Alexandre e Henrique são dois amigos que adoram curiosidades a respeito de palavras. Em uma de suas conversas, eles estavam tentando advinhar qual a maior palavra do inglês onde não se repetem letras. A maior palavra que Alexandre conseguiu pensar foi *uncopyrightable* e Henrique *dermatoglyphics*, ambas com 15 letras distintas.

Depois que nenhum dos dois conseguiu pensar em nenhuma palavra do inglês que tenha mais de 15 letras distintas, eles resolveram procurar mais sobre o assunto na internet. Infelizmente, não é algo que as pessoas costumam discutir muito sobre, então foram poucos os resultados obtidos. Porém, um dos resultados encontrados interessou bastante a ambos. Um site reuniu várias sentenças de diversos idiomas, onde cada sentença contém a maior palavra desse idioma. Super empolgados eles clicaram no link para ver o conteúdo. Para a decepção deles, a página estava com um problema de formatação, todas as palavras das sentenças estavam concatenadas.

Alexandre e Henrique ficaram muito tristes pois com esse problema eles perderam a informação que estavam buscando. Mas eles não haviam desistido, ainda restava uma coisa que podia ser feita. Sabendo da sua habilidade como programador em resolver problemas, eles pediram a sua ajuda para encontrar o tamanho da maior subsequência contígua de caracteres distintos na sentença que foi concatenada. Com isso eles poderiam descobrir se esse resultado encontrado por você seria alguma palavra válida de algum idioma.

Entrada

A entrada possui duas linhas. A primeira linha contém apenas um inteiro N, o número de caracteres na sentença concatenada. Na linha seguinte, temos apenas uma string sem espaços representando as sentenças retiradas do site encontrado por Alexandre e Henrique

Saída

A saída contém apenas um número, que é o tamanho da maior subsequência contígua de caracteres distintos na string de entrada.

Limites

$1 \leq N \leq 10^6$

Observações

A sentença contém apenas letras minúsculas e dígitos.

Exemplos

Entrada
19 42istheanseroflife
Saída
10

Entrada
28 uncopyrightabledomaincontent
Saída
16

Entrada
49 dermatoglyphicsisthescientificstudyoffingerprints
Saída
15

Problema G. Gordura Magra

Arquivo: `gordura.c`, `gordura.cpp` ou `gordura.java`
Limite de tempo: 3 segundos

Graco é um renomado químico que trabalha para a indústria alimentícia. Seu trabalho atual consiste em descobrir um novo tipo de gordura que possa ser usada para melhorar a textura dos alimentos. Porém, para obter alimentos light, essa gordura não deve ser absorvida pelo organismo humano.

Em sua pesquisa, Graco descobriu que a força das interações entre os átomos é um fator relevante na alteração da textura dos alimentos. Cada ligação entre um par de átomos tem uma força W, que é igual ao efeito direto de um átomo no outro. Como pode haver vários "caminhos" ligando, direta ou indiretamente, dois átomos, inclusive passando por um mesmo átomo várias vezes, é difícil determinar o quanto cada átomo realmente afeta os outros. Por algum motivo obscuro, Graco chegou à seguinte conclusão: um átomo A afeta um átomo B de forma igual ao somatório das forças dos "caminhos" que os ligam, e a força total de um caminho é igual ao produto das forças das ligações que o formam. Como possivelmente existem infinitos caminhos, considerando-se loops, Graco não conseguiu implementar o cálculo do efeito. Assim, Graco pediu para você implementar um programa que, dada a molécula de gordura, calcule o efeito considerando somente caminhos de até X ligações de comprimento.

Entrada

A primeira linha de cada teste é composta por três inteiros, N , M e X onde N representa os átomos, M ($1 \leq M \leq N^2$) representa as interações entre eles e X o número máximo de ligações permitido nos caminhos. A seguir terá M linhas, cada linha será composta por três inteiros U, V ($1 \leq U, V \leq N$), W, onde U e V representam os índices dos átomos e W é a força da ligação entre eles. A próxima linha contém um inteiro T, que representa o número de consultas a serem feitas, terá dois inteiros I e J em cada uma das T linhas subsequentes.

Saída

Para cada uma das T consultas feitas, o seu programa deverá imprimir o efeito total que átomo I faz no atomo J.

Limites

$2 \leq N \leq 20$

$1 \leq X \leq 8$

$1 \leq W \leq 10$

$1 \leq T \leq N * (N-1)/2$

Exemplos

Entrada	Saída
3 3 2 1 2 5 1 3 5 2 3 5 1 1 3	30

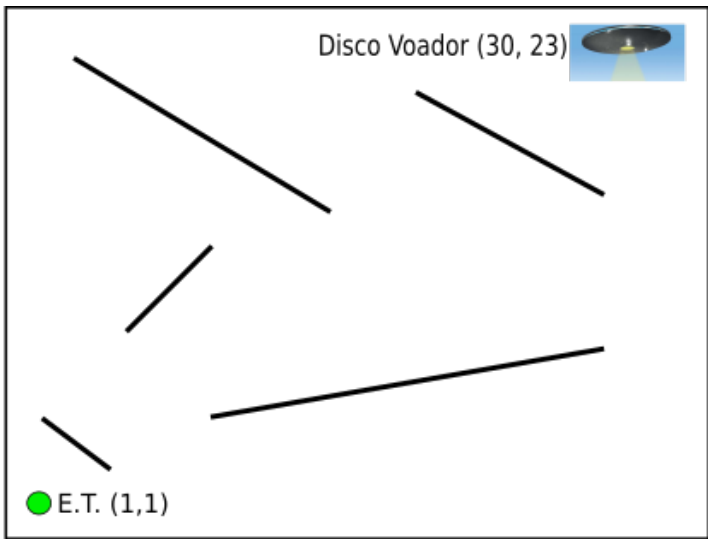
Entrada	Saída
5 5 3 1 2 5 1 5 5 2 3 5 3 4 5 4 5 5 3 1 2 1 4 3 1	380 150 150

Problema H. A Fuga Do E.T.

Arquivo:	`fuga.c`, `fuga.cpp` ou `fuga.java`
Limite de tempo:	1 segundo

Por vários anos, um extra-terrestre vistou a cidade de Varginha. Em cada aparição do E.T., eram registrados aumentos alarmentes no número de roubos de vacas e no número de abduções de pessoas. Cançados de sofrer prejuízos com o tão amado visitante, os moradores resolveram dificultar sua fuga. O prefeito da cidade resolveu, portanto, criar uma competição para descobrir a melhor maneira de ter sucesso nessa árdua tarefa.

A melhor solução foi proposta por Gabriel que desenvolveu muros anti-aliens. Esses muros foram projetados com uma tecnologia avançada que não permite que o E.T. escale, pulverize ou voe com bicicletas voadoras por cima do muro. Como nenhuma tecnologia é perfeita, infelizmente, eles só podiam ser construídos em linha reta. Com o projeto nas mãos e um mapa com os pontos onde era possível construir os muros, os moradores criaram um sistema de muros para tentar capturar o alien.



Em um belo dia, o E.T. retornou à Varginha para reabastecer seu estoque de vacas e foi surpreendido com esse sistema de muros. Desesperado, ele correu até sua casa e pediu que voce calculasse o menor caminho até sua espaçonave. Mediante sua paixão por criaturas intergaláticas, voce foi obrigado a ajudar.

Entrada

A entrada é composta por múltiplas linhas. Na primeira linha, temos 5 inteiros: X_a, Y_a, X_b, Y_b e N. Os dois primeiros números X_a, Y_a representam as coordenadas atuais do alienígena. Os dois números seguintes X_b, Y_b representam as coordenadas do disco voador. Por fim, N representa a quantidade paredes construida pelos moradores de Varginha. As N linhas seguintes são compostas por 4 inteiros X_0, Y_0, X_1e Y_1, que são, respectivamente, as coordenadas de início e fim de cada parede.

Saída

A saida deverá conter apenas uma linha contendo um valor D, com exatamente duas casas decimais de precisão, representando o menor distância que o E.T. tem que percorrer.

Limites

- $1 \leq N \leq 100$.
- Todas as coordenadas estão entre -10^4 e 10^4, inclusive.

Observações

- Em nenhum ponto, uma parede toca outra.

Exemplos

Entrada	Saída
1 1 30 23 3 1 2 2 1 4 6 6 4 15 16 15 25	36.61

Entrada	Saída
0 0 0 6 1 -1 3 1 3	6.32

Problema I. Fila

Arquivo: `fila.c`, `fila.cpp` ou `fila.java`
Limite de tempo: 4 segundos

É hora do recreio na escola primária que João frequenta. Todas as crianças estão espalhadas pelo pátio, brincando. De repente, chega a diretora da escola para a hora cívica. Nesse momento, João sabe que ele e seus colegas devem formar uma fila no pátio o mais rápido possível, pois a diretora não é muito paciente.

João é muito inteligente e estudioso. Enquanto seus colegas estão brincando de amarelinha, ele estuda geometria euclidiana. Ele sabe, por exemplo, que pode modelar o pátio da escola como um plano, cada criança como um ponto nesse plano, e a fila como uma reta. Apesar da diretora ser temida pelos alunos, ela não é muito exigente quanto a aparência da fila. Ela só exige que todos os alunos estejam em uma linha reta, independentemente do espaçamento entre eles.

Para simplificar as coisas, João assumiu que cada uma das crianças é capaz de se mover com velocidade constante de um metro por segundo. Assumindo que todas as crianças cooperem, João quer determinar qual o menor tempo necessário para formar uma fila com todas as crianças. Ele anotou a posição de cada uma das crianças e tentou resolver o problema usando papel e lápis, mas logo percebeu que seria útil usar um computador. Apesar de ser muito esperto, João ainda não sabe programar e vai aprender só no próximo mês. Por isso, ele pediu a sua ajuda. Dadas as coordenadas de cada criança, determine o menor tempo necessário para elas formarem uma fila.

Entrada

A primeira linha da entrada contém um inteiro N, indicando o número de crianças. Cada uma das N linhas seguintes contém dois inteiros X_i e Y_i, indicando as coordenadas das crianças. O valor de cada coordenada é medido em metros.

Saída

A saída deve conter uma única linha, contendo um único número com exatamente duas casas decimais. Esse número deve corresponder ao tempo mínimo necessário para a formação da fila.

Limites

$3 \leq N \leq 100$

$0 \leq X_i \leq 1000$

$0 \leq Y_i \leq 1000$

Exemplos

Entrada	**Saída**
3 0 1 0 5 0 7	0.00

Entrada	**Saída**
4 0 0 1 0 1 1 0 1	0.50

Problema J. Análise de Sentimento

Arquivo:	`sentimento.c`, `sentimento.cpp` ou `sentimento.java`
Limite de tempo:	1 segundo

Análise de sentimento é o processo de analisar um texto e determinar se ele expressa uma opinião positiva ou negativa sobre um assunto. A cada dia, mais e mais pessoas postam suas opiniões publicamente online. Devido a isso, essa análise vem se tornando frequente entre empresas, políticos, e, bem, qualquer um que queira estimar qual é a opinião popular sobre algum assunto.

Sua tarefa aqui é escrever um analisador de sentimento simples. Você tem acesso à uma lista de palavras que expressam uma opinião positiva (por exemplo, "bom", "fantástico", "gostei", ...) e outra lista de palavras que expressam uma opinião negativa (exemplos: "terrível", "péssimo", "desolador", ...). Uma mensagem representa uma opinião positiva se ela possui mais palavras da lista positiva do que da negativa. Se ela possui mais palavras da lista negativa do que da positiva, então ela representa uma opinião negativa. Se o número de palavras da mensagem que estão em cada uma das listas é igual, então a mensagem é neutra.

São dados o nome de um produto, e uma lista de mensagens extraídas da internet. Calcule quantas das mensagens da lista que contem o nome do produto são positivas, neutras e negativas.

Entrada

A primeira linha da entrada contém o nome do produto, que pode ser composto por letras e números, e possui no máximo 50 caracteres.

A segunda linha contém um único inteiro P, que é o número de palavras na lista positiva. Em seguida, há P linhas, cada uma das quais representa uma palavra que expressa opinião positiva. Essas palavras possuem apenas letras (minúsculas ou maiúsculas) e não possuem mais do que 50 caracteres cada.

Em seguida, há uma linha contendo um único inteiro N, que é o número de palavras na lista negativa. Após essa linha, há N outras linhas, cada uma das quais representa uma palavra que expressa opinião negativa. De novo, essas palavras possuem apenas letras (minúsculas ou maiúsculas) e não possuem mais do que 50 caracteres cada.

Por fim, há uma linha contendo um único inteiro M, seguida por M linhas que representam mensagens coletadas da Internet. Cada uma dessas mensagens possui letras, dígitos e espaços (mas não pontuação), e nenhuma delas possui mais do que 200 caracteres.

Saída

Imprima uma linha contendo três inteiros N_p, N_{meh} e N_n, que são, respectivamente, o número de mensagens que mencionam o produto e expressam uma opinião positiva, o número de mensagens que mencionam o produto e expressam uma opinião neutra, e o número de mensagens que mencionam o produto e expressam uma opinião negativa.

Observações

- Uma mensagem contém uma palavra se essa palavra aparece em algum ponto da mensagem, independentemente da caixa (maiúsculas ou minúsculas), e não está contida inteiramente dentro de outra palavra. Por exemplo, a mensagem "`secret enchanted brocolli forest é a música com o nome mais absurdo da história`" contém a palavra "`AbSuRDo`", mas não contém a palavra "`surdo`" — essa última aparece na mensagem, mas só como uma sub-parte de outra palavra maior.
- O nome do produto nunca está contido na lista de palavras positivas nem na lista de palavras negativas.
- Não existe nenhuma palavra que esteja tanto na lista de palavras positivas quanto na de negativas.
- As palavras de uma mensagem são sempre separadas por apenas um espaço em branco.

Limites

$0 < P \leq 100$
$0 < N \leq 100$
$0 < M \leq 10^5$

Exemplos

Entrada
lol 2 sensacional bom 5 chato ruim lento bobo feio 6 LoL eh muito ruim LOL eh chato bobo e feio LOl eh sensacional LOL eh meh Prefiro dota Lol eh tao polemico quanto mamilos
Saída
1 2 2

Entrada
pagode 2 bom legal 4 terrivel mal pessimo pior 3 Pagode eh a raiz de todo mal do universo Pagode pior estilo musical do Brasil Viva o Rock
Saída
0 0 2

Problema K. War

Arquivo: `war.c`, `war.cpp` ou `war.java`
Limite de tempo: 3 segundos

Num conhecido jogo de estratégia, Alexandre e Dilson se enfrentam simulando guerras entre pares de territórios em que Alexandre conta com um conjunto de N_A exércitos de ataque e Dilson conta com N_D exércitos de defesa. A guerra se dá por meio de uma sequência de batalhas. Cada batalha faz com que pelo menos um dos dois jogadores perca exércitos segundo regras que explicaremos a seguir. As batalhas prosseguem até que Alexandre tenha um único exército ou até que o Dilson seja completamente dizimado e fique sem exército nenhum.

Uma batalha começa com a escolha do número de exércitos que cada jogador irá utilizar. Denotaremos a quantidade de exércitos escolhidos pelo atacante Alexandre para uma dada batalha como n_A e a quantidade de exércitos escolhidos pelo defensor Dilson para uma dada batalha como n_D. O atacante é o primeiro a escolher, seguido do defensor. O atacante pode escolher até 3 exércitos, sendo que pelo menos um exército precisa permanecer em seu território. Por exemplo: se o atacante dispor de 3 exércitos em um dado momento, o número máximo de exércitos que podem ser escolhidos para a batalha é 2. Com 4 ou mais exércitos, o máximo é 3. O defensor também está limitado a 3 exércitos, mas ele não precisa manter um exército em seu próprio território. Sendo assim, numa situação em que o defensor tenha 3 exércitos, o máximo permitido é 3. Com dois exércitos, o máximo é 2; já com 4 ou mais, o máximo permitido permanece 3.

Após a escolha da quantidade de exércitos a serem utilizados por cada jogador na batalha, n_A dados de ataque e n_D dados de defesa com seis faces numeradas de 1 até 6 cada são jogados. Os dados jogados por Alexandre são ordenados, assim como os dados de Dilson. A seguir, os $\min(n_A, n_D)$ maiores valores obtidos pelos jogadores são comparados par a par. Para cada comparação em que o número do atacante for maior que o do defensor, a defesa perde um exército. Já quando o número da defesa for maior, o atacante perde um exército. Finalmente, em caso de empate, o atacante perde um exército. Por exemplo: numa batalha em que Alexandre escolheu usar $n_A = 3$ exércitos, Dilson escolheu usar $n_D = 2$ exércitos e os valores obtidos foram $\{1, 4, 3\}$ e $\{2, 3\}$, Dilson perde dois exércitos (pois $4 > 3$ e $3 > 2$). Já numa batalha de 3 contra 3 em que os valores obtidos foram $\{5, 3, 1\}$ para o ataque e $\{4, 4, 1\}$ para a defesa, o ataque perde dois exércitos e a defesa perde 1.

Alexandre tem perdido muitas guerras para Dilson. Tendo isso em vista, ele pediu que você escrevesse um programa que recebe as quantidades de exércitos dos dois jogadores e retorna a probabilidade de que Alexandre ganhe uma guerra supondo que *ambos joguem de maneira ótima*.

Entrada

A entrada contém dois inteiros N_A e N_D separados por um espaço em branco.

Saída

Imprima um único número p_A com quatro casas decimais de precisão seguido por uma quebra de linha. Esse número é a probabilidade de que Alexandre ganhe a guerra.

Limites

$1 \leq N_A \leq 10000$ $1 \leq N_D \leq 10000$

Exemplos

Entrada	Saída
1 1	0.4167

Entrada	Saída
5 4	0.3406

Problema L. Dividindo a Conta

Arquivo: `conta.c`, `conta.cpp` ou `conta.java`
Limite de tempo: 1 segundo

Um grupo de amigos foi a um restaurante, onde comeram, beberam, e discutiram qual linguagem de programação era a melhor. Depois de serem rechaçados por preferirem Java, alguns dos amigos foram embora, mas não sem antes deixar, cada um, uma contribuição para o pagamento da conta. Ao final da noitada, quando todos os restantes concordaram que a melhor linguagem seria uma mistura de Erlang, C, e Ada, os demais amigos dividiram o restante da conta igualmente. Como não poderia deixar de ser, sobrou para você implementar um programa que calcule o quanto coube a cada um pagar.

Entrada

A entrada possui apenas uma linha contendo os seguintes valores, nesta ordem: inteiro n, o número total de amigos que foi ao restaurante; inteiro m, o número de amigos que foi embora mais cedo; real c, o valor total da conta, em Real e centavos de Real; real p, o quanto cada um dos m amigos deixou para pagar a conta, em Real e centavos de Real.

Saída

Imprima uma linha contendo o quanto cada um dos $n - m$ amigos que sobraram no fim da festa devem pagar do próprio bolso, sendo que cada um destes deve pagar exatamente o mesmo valor.

Observações

A menor unidade monetária utilizada é 1 centavo de Real. A saída sempre deve ter exatamente 2 casas decimais de precisão.

Limites

$0 \leq m < n \leq 10^5$

$0 \leq c \leq 10^6$

$0 \leq P \leq 10^6$

Exemplos

Entrada	Saída
10 5 100.00 20.00	0.00

Os ”javaneses”pagaram toda a conta.

Entrada	Saída
5 1 27.00 6.00	5.25

Sobraram 21 Reais para serem divididos por 4 amigos.

Entrada	Saída
5 2 30.0 5.0	6.67

Sobraram 20 Reais para serem divididos por 3 amigos.