A Inconfidente

1^a Seletiva Interna – 2012/1

Sevidor BOCA:

`http://10.20.107.205/boca/` (acesso interno)

`http://200.19.107.205/boca/` (acesso externo)

Organização e Realização:

Claudio Cesar de Sá (coordenação geral), Lucas Negri (coordenação técnica), Alexandre Gonçalves Silva, Roberto Silvio Ubertino Rosso Jr., André Luiz Guedes (revisão técnica – Dinf/UFPr), Gilmário Barbosa Santos, Gian Ricardo Berkenbrok, Rogério Eduardo da Silva

Lembretes:

- Aos *javaneiros*: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido**. Ex: **classe petrus**, nome do arquivo **petrus.java**;
- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão;
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos.

Patrocinador e Agradecimentos

- YoungArts;
- DCC/UDESC;
- Rutes, Zanatta e Weskley que tiveram muito empenho neste início de ano;
- Alguns, muitos outros anônimos.

A Inconfidente

1ª Seletiva Interna da UDESC

20 de abril de 2012

Conteúdo

1 Problema A: As Roupas de Jean and Joe

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

O Problema

Jean e Joe formam um casal bastante desorganizado e de um estilo meio jeca de se vestir. Seus 2 filhos, Jane e James também se vestem nesse mesmo estilo.

A mãe de Jean veio visitá-lo e ficou horrorizada com o estado desordenado da casa e procurou colaborar formando uma pilha de roupas para cada uma das 4 pessoas na casa, fez isso olhando para o tamanho de cada item. O problema é que infelizmente, por vezes, o tamanho havia sido cortada ou não era legível, essas roupas (sem identificação de tamanho) ela colocou em uma pilha separada.

Se o tamanho for 'M' ou 'L', então é Joe.
Se o tamanho for 'S', então é James.
Se é 12 ou superior, então é de Jean.
Se é menor do que 12 é de Jane.

A mãe de Jean faz visitas freqüentes e tem que passar por este processo a cada visita. Sua tarefa nesse problema é escrever um programa que irá ajudá-la.

Entrada

A entrada consiste de dados para um número de visitas da mãe de Jean. A primeira linha de cada visita será constituído por um número inteiro, N ($0 < N <= 50$), que é o número total de roupas encontradas espalhadas pela casa. Depois segue-se N linhas, cada um representando o tamanho de um item de vestuário, ou um 'X' se ele estiver ausente ou ilegível. Os tamanhos poderão ser: um número de 2 dígitos ou uma letra 'S', 'M' ou 'L'.

A entrada é terminada quando o número de roupas espalhadas, N é 0. Não processar esta linha.

Saída

A saída consiste de uma linha para cada visita. A linha irá conter cinco números, cada um separado por um espaço. Os números representam a quantidade de roupas pertencentes a Joe, Jean, Jane e James, respectivamente, na visita atual. O número final é a quantidade de roupas atribuída a ninguém. Se não houver nenhuma roupa em uma pilha, o número 0 deve ser mostrado.

Exemplo de Entrada

```
8
M
12
X
14
10
L
S
S
0
```

Exemplos de Saída

```
2 2 1 2 1
```

```
Explicando:
Joe tem 2 (M e L)
Jean tem 2 (12 e 14)
Jane tem 1 (10)
James tem 2 (S e S)
Existe um X nao legivel
```

2 Problema B: Sequências

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

Uma sequência aritmética apresenta um primeiro número, e então uma sequência de outros números com diferença fixa entre eles. Por exemplo: `3, 5, 7, 9` é uma sequência aritmética pois ela tem como primeiro número o `3`. E então, cada um de seus termos da sequência é formado pela adição do valor `2` ao termo anterior. Ou seja, os termos são diferentes por `2`. O `3` é o primeiro termo ou termo 1, o `9` o quarto termo ou o termo 4. Dado um número de partida, uma diferença e um valor, seu programa deve indicar se este último número pode tomar parte da sequência. Se sim, a saída indicará qual o número do termo nesta sequência, e se não, a saída deverá ter a letra `X`.

Entrada

A entrada consistirá de um número de linhas, onde cada linha tem 3 números separados por espaços em branco. O primeiro número é um inteiro que é o primeiro termo da sequência. O segundo é a diferença, o qual será um inteiro diferente de zero. O terceiro termo é o valor o qual voce necessitará testar para determinar se este número pode fazer parte da sequência ou não. A entrada é finalizada por um valor zero para cada um dos tres números.

Saída

A saída consistirá de uma linha para cada entrada. Ela terá como o valor o número que indica a qual o termo ele é, ou `X` se o número não fizer parte da sequência.

Exemplo de Entrada

```
3 2 11
-1 -3  -8
0 0 0
```

Exemplo de Saída

```
5
X
```

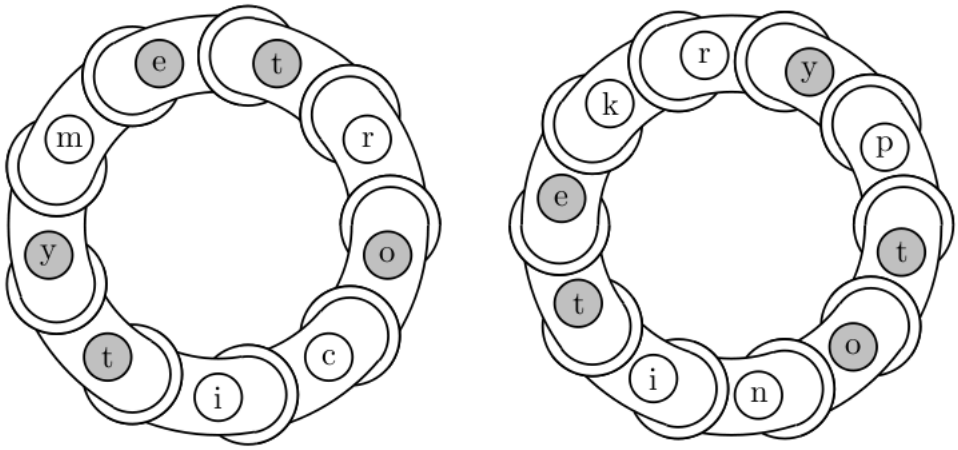
Explicando:

- No primeiro caso, o 11 é o quinto termo da sequência, ou o termo 5;
- No segundo caso, o `-8` não faz parte da sequência.

3 Problema C: Bracelets

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

Finally, Megamind has devised the perfect plan to take down his arch-nemesis, Metro Man! Megamind has designed a pair of circular power bracelets to be worn on his left and right wrists. On each bracelet, he has inscribed a sequence of magical glyphs (i.e., symbols); each activated glyph augments Megamind's strength by the might of one grizzly bear! However, there's a catch: the bracelets only work when the subsequences of glyphs activated on each bracelet are identical. For example, given a pair of bracelets whose glyphs are represented by the strings "metrocity" and "kryptonite", then the optimal activation of glyphs would give Megamind the power of 10 grizzly bears:



(a) Left bracelet (b) Right bracelet

Figura 1: Megamind's power bracelets

On the first bracelet, the letters "etoty" are activated in clockwise order; the same letters are activated in counterclockwise order on the second bracelet. Generally, the ordering of the letters is important, but the orientation of the activated subsequence on each bracelet (i.e., clockwise or counterclockwise) may or may not be the same—and don't forget that the bracelets are circular! Help Megamind defeat Metro Man by determining the optimal subsequences of glyphs needed to activate his bracelets.

Input Specification

The input file will contain at most 50 test cases (including at most 5 "large" test cases). Each test case is given by a single line containing a space-separated pair of strings s and t, corresponding to the sequences of glyphs on Megamind's left and right power bracelets, respectively. Each string will consist of only lowercase letters ('a'-'z'). The length of each input string will be between 1 and 100 characters, inclusive.

Output Specification

For each input test case, print the maximum power (in units of grizzly bears) that Megamind will be able to achieve by activating glyphs on his bracelets.

Sample Input

```
metrocity kryptonite
megamind agemdnim
metroman manmetro
megamindandmetroman metromanandmegamind
```

Sample Output

```
10
16
16
32
```

PS: limit time 08 seconds

4 Problema D: Os Brinquedos de Maria

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

Maria, de 5 anos, tem vários brinquedos de pelúcia. Eles devem ser mantidos no quarto dela, mas nem sempre todos são trazidos de volta a cada noite. Quando ela vai para a cama à noite, ela coloca os brinquedos de pelúcia que estão em seu quarto na cama ao lado da dela. Ela gosta de dar a cada brinquedo um amigo para a noite e de colocá-los em pares. Às vezes não há um número par de brinquedos, então ela pode ter que fazer um grupo de 3. Sua tarefa é escrever um programa que apresentará um relatório sobre a disposição dos brinquedos de Maria em uma noite específica.

Entrada

A entrada será um número de linhas, com cada linha representando uma noite na casa de Maria. Cada linha terá 2 números inteiros, separados por um espaço. O primeiro número é quantos brinquedos de pelúcia ela possui no momento. O segundo número é o número de brinquedos deixados pela casa que não foram trazidos de volta para o quarto de Maria naquela noite. A última linha será 0 0. Não processar esta linha!

Saída

A saída será composta de 2 números separados por um espaço para cada noite. O primeiro representa quantos pares existem e o segundo será 0 ou um 1. Um zero significa não existem grupos de 3 e um 1 representa que existia um grupo de 3.

Exemplo de Entrada

```
7 3
6 3
5 4
0 0
```

Exemplo de Saída

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

Explicando os testes:

- Apenas 4 brinquedos (7 menos 3) estão emparelhados, formando 2 pares.;
- 3 brinquedos formam 1 grupo de 3 (e nenhum par);
- Existe apenas 1 brinquedo.

5 Problema E: O Equilíbrio do Mundo

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

O professor de LFM de sua universidade está procurando dar um sentido prático as gramáticas livre-de-contexto (as *glc*). Neste dia de aula sobre as *glcs*, ele acordou inspirado para sua aula de LFM:

> *"O mundo é quase um equilíbrio constante. Se de um lado temos o positivo, do outro lado o negativo, é a luz versus sombra, . . . e se há o abre parênteses (um delimitador) à esquerda, do outro lado temos o fecha parênteses à direita, e ai vai. . . . "*

Nesta história de delimitadores e equilíbrio, sua missão é escrever um programa que ajude o professor de LFM de sua universidade a avaliar se uma frase (*string*) está equilibrada, no que diz a respeito aos delimitadores neste equilíbrio de mundo.

Uma seqüência/frase dada ao programa pode ter dois tipos de delimitadores, os parênteses (*arrendodados*) ("( )") e colchetes (*quadrados*) ("[ ]"). Uma seqüência é dita ser equilibrada, se e somente se, as seguintes condições forem válidas:

1. Para cada parênteses à esquerda ("("), há um correspondente à direita (")") na parte subsequente da cadeia/frase;
2. Para cada colchete à esquerda ("["), há um colchete correspondente à direita ("]") na parte subsequente da cadeia/frase;
3. Para cada um dois delimitadores à direita, há um delimitador correspondente a ele à esquerda;
4. As correspondências dos delimitadores tem que ser um para um, ou seja, nunca um delimitador único corresponde a dois ou mais outros;
5. Para cada par de delimitador correspondente à direita e à esquerda, a substring entre eles é equilibrada.

Entrada

A entrada consiste de uma ou mais linhas, cada uma das quais sendo um conjunto de dados. Um conjunto de dados é uma seqüência caracteres que consiste de uma frase em inglês, caracteres de espaço em branco e os dois tipos de delimitadores, "()"e "[ ]", finalizado por um ponto. Você pode assumir que cada linha tenha até 100 caracteres. Uma linha com um único ponto indica o final da entrada. Este deverá estar na primeira coluna. Caso contrário, pode ser uma frase vazia (ver o último caso do teste), o qual é "equilibrado", pois é vazio!

Saída

Para cada conjunto de dados, indicar se a cadeia é equilibrada com uma saída com "`yes`", ou "`no`", em caso contrário, em uma linha. A linha não pode conter quaisquer caracteres extras na saída.

Exemplo de Entrada

```
So when I die (the [first] I will see in (heaven) is a score list).
[ first in ] ( first out ).
Half Moon tonight (At least it is better than no Moon at all].
A rope may form )( a trail in a maze.
Help( I[m being held prisoner in a fortune cookie factory)].
([ (([( [ ] ) ( ) (( ))] )) ]).
 .
.
```

Exemplo de Saída

```
yes
yes
no
no
no
yes
yes
```

6 Problema F: Alvo

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

Um jornal internacional, disponível na Quiprocó, tem um quebra-cabeça chamado Alvo. O quebra-cabeça apresenta um tabuleiro de 9 letras, do qual os participante são convidados a construírem palavras de, pelo menos, 4 letras. Pelo menos uma palavra deve conter todas as nove letras. A letra central no tabuleiro deve estar presente em cada palavra.

```
M   N   I
O  >P<  A
C   R   A
```

No exemplo mostrado, todas as palavras devem conter a letra P. A solução dada pelo jornal para esse quebra-cabeça é: apian apron camp campion capon carp corpa corp cramp crampon crimp crop napa orpin pain pair panic panoramic para piano pica pion pram prana prim prima primo prom ramp rampion romp

Nesse problema, você receberá uma dada solução e é solicitado a recriar o quebra-cabeça. Em tua resposta, as letras do quebra-cabeça devem estar em ordem alfabética, exceto pela letra central (P no quebra-cabeça exemplo) que deve estar em quinto lugar na lista (e.g. no meio). O quebra-cabeça acima deve ser mostrado como

```
A A C I P M N O R
```

Como dito, a letra requerida P está localizada no meio.

Entrada

A entrada consiste de vários quebra-cabeças. Cada quebra-cabeça começa com um inteiro em particular, N ($2 < N \leq 50$), que representa o número de palavras na solução do quebra-cabeça. A última linha da entrada é o caso onde $N = 0$. Esse quebra-cabeça não deve ser processado.

A entrada para cada quebra-cabeça irá conter N palavras, cada uma em linha separada. Palavras serão inteiramente em caixa baixa e irá conter entre 4 e 9 letras. Pelo menos uma palavra em cada quebra-cabeça irá conter 9 letras. Todas as palavras no quebra-cabeça irão conter somente letras encontradas na palavras de 9 letras, onde nenhuma letra ocorrerá mais vezes em uma única palavra do que ocorre na palavra de 9 letras. Haverá somente uma letra do alfabeto que ocorrerá em todas as palavras.

Saída

A saída será uma linha com o quebra-cabeça. Essa linha irá conter as 9 letras do quebra-cabeça, em caixa ALTA e separados por espaços, em ordem alfabética exceto na letra requerida, a qual deve estar localizada na quinta posição desconsiderando a posição alfabética.

Exemplo de Entrada

```
31
apian
apron
camp
campion
capon
carp
corpa
corp
cramp
crampon
crimp
crop
napa
orpin
pain
pair
panic
panoramic
para
piano
pica
pion
pram
prana
prim
prima
primo
prom
ramp
rampion
romp
0
```

Exemplo de Saída

```
A A C I P M N O R
```

7 Problema G: E Então, Quantos São?

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

Para o Sr. Solitário, que é um famoso e solitário criador de jogos, uma nova ideia ocorre todos os dias. Seu novo jogo requer discos de várias cores e tamanhos.

Para começar, todos os discos são distribuídos aleatoriamente em torno do centro de uma mesa. Durante o jogo, você pode remover um par de discos da mesma cor, mas somente se em nenhum deles tiver qualquer outro disco em cima. Note que um disco não deve ser considerado em posição acima de outro quando o estiver tangenciando externamente.

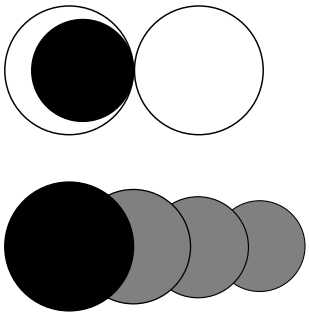


Figura 2: Sete discos na mesa

Por exemplo, na Figura 2, você só pode remover os dois discos pretos em um primeiro momento. Esta remoção, por sua vez, torna possível a remoção dos dois brancos. Por outro lado, os discos cinzas nunca poderão ser removidos.

Você foi requesitado a escrever um programa de computador que, para discos com posições iniciais conhecidas, tamanhos e cores, calcule o número máximo de discos que podem ser removidos.

Entrada

A entrada consiste de vários conjuntos de dados, cada um com o seguinte formato, representando o estado de um jogo depois de todos os discos serem dispostos sobre a mesa.

n
$x_1\ y_1\ r_1\ c_1$
$x_2\ y_2\ r_2\ c_2$
$\vdots$
$x_n\ y_n\ r_n\ c_n$

A primeira linha é constituída por um número inteiro positivo n representando o número de discos. As n linhas seguintes, cada uma contendo 4 inteiros separados por um único espaço, representam as posições iniciais, tamanhos e as cores dos n discos, da seguinte maneira:

- (x_i, y_i), r_i e c_i são as coordenadas x e y do centro, o raio e o número de índice de cor, respectivamente, do i-ésimo disco, e

- sempre que o i-ésimo disco é colocado em cima do j-ésimo disco, $i < j$ deve ser satisfeito.

Pode assumir que cada número de índice de cor está entre 1 e 4, inclusive, e no máximo 6 discos em um conjunto de dados são da mesma cor. Você também pode assumir que o as coordenadas x e y do centro de cada disco estão entre 0 e 100, inclusive, e o raio entre 1 e 100, inclusive.

O fim da entrada é indicado por um único zero.

Saída

Para cada conjunto de dados, imprimir uma linha contendo um número inteiro que indica o número máximo de discos que podem ser removidos.

Exemplo de Entrada

```
4
0 0 50 1
0 0 50 2
100 0 50 1
0 0 100 2
7
12 40 8 1
10 40 10 2
30 40 10 2
10 10 10 1
20 10 9 3
30 10 8 3
40 10 7 3
2
0 0 100 1
100 32 5 1
0
```

Exemplo de Saída

```
2
4
0
```

8 Problema H: Rede Congestionada

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

Dada uma rede de computadores conectados (comunicação bidirecional) queremos encontrar dois nós diferentes u e v tal que possamos maximizar o congestionamento entre u e v com um tipo de vírus enviado continuamente entre este par. Definimos o nível de congestionamento como o número máximo de caminhos **arco–disjuntos** entre os nós u e v.

Por exemplo, o circuito mostrado na figura 3 tem três diferentes caminhos (arco–disjuntos) entre os nós 0 e 6 de tal forma que cada arco do caminho é apenas utilizado uma vez como parte do mesmo caminho. Os três caminhos na figura 3, estão evidenciados por tracejados, pontilhados e linha continua. Note que dois caminhos quaisquer são autorizados a passar por um mesmo nó, no caso deste exemplo, é o nó 7. Nenhum outro par de nós proporciona um nível maior congestionamento.

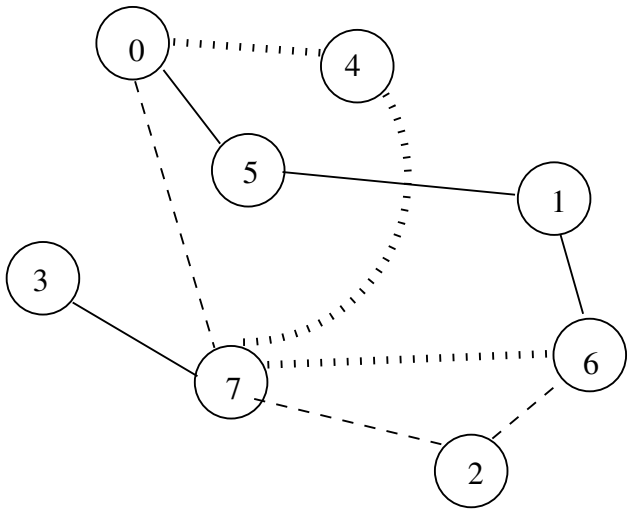


Figura 3: Primeiro exemplo

Entrada

A entrada dada é uma seqüência de redes de computadores conectados, cada um com n nós, $n \leq 40$, rotulado $\{0, 1, ..., n-1\}$. O último caso da entrada será seguido por um rede com $n = 0$ nodos, que não deve ser processada. A especificação para uma rede de computadores, será a seguinte:

- Na primeira linha contém um valor não-negativo n definindo o número de nós da redes;
- Em seguida, esta será composta por n linhas de inteiros, separados por espaços, que define a lista de vizinhos imediatos de cada nó;
- Esta sequência é indexada, inicializada, com o nó 0 até o nó $n-1$;
- Espera-se até 2000 casos de teste;
- Ultima linha das entradas contém um 0 (ou a entrada termina com 0).

Saída

Para cada caso, ou topologia de rede, a saída será um inteiro que corresponde ao nível máximo de congestionamento possível para algum par de nós da rede.

Observação: a figura 3 corresponde ao primeiro caso do exemplo que se segue.

Exemplo de Entrada

```
8
4 5 7
5 6
6 7
7
0 7
0 1
1 2 7
0 2 3 4 6
4
1 2
0 2
3 0 1
2
0
```

Exemplo de Saída

```
3
2
```

9 Problema I: Vendas

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

Uma grande grife, Bolsas Coloridas & Chiques – (**BCC**) quer medir o desempenho dos seus vendedores. Como um passo inicial, esta pretende contar quantas vendas cada vendedor realizou, a partir do registro de vendas de cada filial. Para verificar disparidades entre cada filial, pretende-se calcular a maior diferença de vendas entre dois vendedores.

Como estagiário de TI desta grife, sua tarefa é analisar os registros de vendas de cada filial e reportar a maior diferença entre o número de vendas dos vendedores.

Tarefa

Escreva um programa que calcule, para cada filial, a maior diferença entre o número de vendas dos vendedores.

Entrada

A entrada contém vários conjuntos de testes, que devem ser lidos do *dispositivo de entrada padrão*.

Cada caso de teste contém duas linhas. A primeira linha contém dois inteiros, s e v que indicam, respectivamente o número de vendas e o número de vendedores da filial. Na segunda linha seguem s inteiros, de 1 até v, separados por um espaço em branco. Nesta lista, o valor do i-ésimo elemento diz qual foi o vendedor que fez a i-ésima venda. A entrada termina com $s = 0$ e $v = 0$.

Limites: $0 \leq s \leq 3000$, $0 \leq v \leq 5000$.

Saída

Para cada caso de teste deve-se imprimir uma linha, na saída padrão, contendo o a maior diferença entre o número de vendas dos vendedores.

Exemplo de Entrada

```
9 5
5 3 2 1 5 2 1 3 2
5 100
100 100 100 100 100
0 0
```

Exemplo de Saída

```
3
5
```

10 Problema J: Aluno Folgado

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

Um determinado aluno da UDESC (muito esperto, porém muito folgado) percebeu que estava investindo muitas horas em estudo, e que queria alocar algumas destas para jogos eletrônicos e construção de foguetes. Como seu pai o proibiu de reprovar em todas as matérias simultaneamente, este decidiu utilizar o tempo restante de forma otimizada, isto é, de forma a ter a menor chance possível de reprovar em todas as disciplinas simultaneamente.

O aluno já construiu uma tabela que relaciona o tempo de estudo e a chance de reprovação em cada disciplina, mas percebeu que alocar suas horas livres de forma ótima é um processo que gasta muitas destas suas horas livres, e pediu para você escrever um programa que faça isto para ele.

Tarefa

Escreva um programa que determine qual é a menor chance do aluno reprovar em todas as disciplinas simultaneamente, quando a alocação das suas horas de estudo é otimizada.

Entrada

A entrada contém vários conjuntos de testes, que devem ser lidos do *dispositivo de entrada padrão*.

A linha inicial contém dois inteiros, h e d, que representam o total de horas disponíveis do aluno e a quantidade de disciplinas. Cada uma das próximas $h + 1$ linhas contém d números reais entre 0 e 1, que representam a chance de reprovação. Cada coluna representa uma disciplina, enquanto que a i-ésima linha representa um gasto de $i - 1$ horas de estudo (a primeira linha significa que não vai estudar enquanto a última significa que vai gastar todo o tempo livre). A entrada termina com $h = 0$ e $d = 0$.

Limites: $0 \leq h \leq 100$, $0 \leq d \leq 100$.

Saída

Para cada caso de teste deve-se imprimir na saída padrão a menor chance do aluno reprovar em todas as disciplinas, representada por um número real entre 0 e 1, arredondada para três casas decimais[1].

Exemplo de Entrada

```
3 3
0.2 0.2 0.2
0.2 0.2 0.2
0.3 0.3 0.3
0.4 0.4 0.1
0 0
```

Exemplo de Saída

```
0.004
```

[1] printf("%.3f", ...);