7^o Treino para Alunos da UFPR

3 de Maio de 2013

Sevidor BOCA:
`http://maratona.c3sl.ufpr.br/boca/`

Organizadores:
Vinicius Kwiecien Ruoso e Ricardo Tavares de Oliveira

Lembretes:

- 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.

A: Jogo do Maior

File: maior.[c|cpp|java|pas]

Og gosta muito de brincar com seus filhos. Seu jogo preferido é o jogo do maior, de autoria própria. Este passatempo (no tempo das cavernas se tinha muito tempo disponível para jogos) é jogado em dupla, Og e um dos seus filhos. O jogo procede da seguinte forma: os dois participantes escolhem um número de rodadas e, a cada rodada, cada participante diz um número de 0 até 10 em voz alta, sendo que o participante que falar o número mais alto ganha um ponto (em caso de empate, ninguém ganha o ponto). No final das rodadas, os pontos são contabilizados e o participante com o maior número de pontos ganha.

Og e seus filhos gostam muito do jogo, mas se perdem na contagem dos pontos. Você conseguirá ajudar Og a verificar a pontuação de uma lista de jogos?

Entrada

A entrada é composta por vários casos de teste (partidas). Cada caso é iniciado com um inteiro N (de 0 até 10) representando o número de rodadas da partida, sendo que o valor 0 representa o final da entrada e não deve ser processado. Cada uma das próximas N linhas contém dois inteiros, A e B, onde A é o número escolhido pelo primeiro jogador e B é o número escolhido pelo segundo jogador ($0 \leq A, B \leq 10$).

Saída

A saída deve ser composta por uma linha por caso de teste, contendo o número de pontos de cada jogador, separados por um espaço.

Exemplo de entrada	**Exemplo de saída**
3 5 3 8 2 5 6 2 5 5 0 0 0	2 1 0 0

B: All Discs Considered

`File: discs.[c|cpp|java|pas]`

Operating systems are large software artefacts composed of many packages, usually distributed on several media, e.g., discs. You probably remember the time when your favourite operating system was delivered on 21 floppy discs, or, a few years later, on 6 CDs. Nowadays, it will be shipped on several DVDs, each containing tens of thousands of packages.

The installation of certain packages may require that other packages have been installed previously. Therefore, if the packages are distributed on the media in an unsuitable way, the installation of the complete system requires you to perform many media changes, provided that there is only one reading device available, e.g., one DVD-ROM drive. Since you have to start the installation somehow, there will of course be one or more packages that can be installed independently of all other packages.

Given a distribution of packages on media and a list of dependences between packages, you have to calculate the minimal number of media changes required to install all packages. For your convenience, you may assume that the operating system comes on exactly 2 DVDs.

Input

The input contains several test cases. Every test case starts with three integers N1, N2, D. You may assume that $1 \leq N_1, N_2 \leq 50000$ and $0 \leq D \leq 100000$. The first DVD contains N_1 packages, identified by the numbers $1, 2, ..., N_1$. The second DVD contains N_2 packages, identified by the numbers $N_1+1, N_1+2, ..., N_1+N_2$. Then follow D dependence specifications, each consisting of two integers x_i, y_i. You may assume that $1 \leq x_i, y_i \leq N_1+N_2$ for $1 \leq i \leq D$. The dependence specification means that the installation of package x_i requires the previous installation of package y_i. You may assume that there are no circular dependences. The last test case is followed by three zeros.

Output

For each test case output on a line the minimal number of DVD changes required to install all packages. By convention, the DVD drive is empty before the installation and the initial insertion of a disc counts as one change. Likewise, the final removal of a disc counts as one change, leaving the DVD drive empty after the installation.

Sample Input	**Sample Output**
3 2 1 1 2 2 2 2 1 3 4 2 2 1 1 1 3 0 0 0	3 4 3

C: O Fantástico Jaspion!

File: jaspion.[c|cpp|java|pas]

Em 1985 estréia na TV Japonesa a série Kyojiu Tokusou Jaspion (Investigador Especial de Monstros Jaspion). A série chega ao Brasil alguns anos depois com o título "O Fantástico Jaspion", e com ela nasce a fantasia de polícia espacial em milhões de brasileirinhos. As crianças saíam da escola, corriam pelas ruas (sem olhar se vinha carro), ligavam a TV e mergulhavam na coragem, exemplo de pessoa, e incontestável sede por justiça do Fantástico Jaspion. O comércio de gibis e as brigas por figurinhas no recreio da escola estavam alcançando números históricos. Até então, tal sentimento só havia sido estimulado com tanta intensidade pelo Chaves e a sua turma! Diante dessa febre inter-galática, o inevitável aconteceu. Os produtores do Jaspion ganharam o Nobel da Paz! Isso mesmo! Os produtores ganharam um Nobel. As histórias do grandioso Jaspion estavam por todo canto. Agora as crianças tinham um belíssimo exemplo para seguir. A paz mundial estava garantida! Não precisávamos mais temer o monstrengo Satan Gos!

No Brasil havia uma criança que adorava as histórias do Jaspion! Antônio Melhorança Capote Valente Junior carinhosamente apelidado de ACM, um menino da zona sul de São Paulo que adorava cantar as músicas do grande herói. Ele era tão fanático que chegou a comprar um dicionário de Japonês-Português e iniciou um trabalho árduo de tradução. Entretanto, o trabalho ficou inacabado! Alguns trechos da canção ainda precisam ser traduzidos. Neste momento você deve estar se perguntando: qual é a minha tarefa neste fabuloso problema? Ok! Antes de falar da sua tarefa, convide seu companheiro de equipe para mergulhar com você no desfecho da história. Para isso, vamos falar mais um pouco sobre o nosso ACM. Ele se formou em Ciência da Computação e hoje trabalha no mesmo escritório que você. Pois é! Você trabalha como programador ao lado dessa figura! Como sabemos que você gosta muito dele, temos certeza que vai aceitar a seguinte tarefa: dado um dicionário Japonês-Português e uma letra de música, escreva um programa que imprima a letra traduzida.

Entrada

A primeira linha de um caso de testes contém um inteiro T que indica o número de instâncias subseqüentes. A primeira linha de cada instância contém dois inteiros M e N ($1 \leq M \leq 1000000$, $1 \leq N \leq 1000$), que representam o número de palavras no dicionário e o número de linhas na letra da música, respectivamente.

Os próximos M pares de linhas contêm as traduções: a primeira linha de cada par contém a palavra em Japonês, e a segunda linha contém a tradução para o Português (que pode ter uma ou mais palavras). Todas as palavras usam apenas letras minúsculas. Cada palavra em Japonês aparece apenas uma vez em cada instância.

As próximas N linhas contêm a letra da música. Cada linha da letra da música é uma lista de palavras separadas por um espaço (todas as palavras consistem apenas de letras minúsculas). Algumas podem estar vazias, mas nenhuma linha possui espaços no início ou no final.

Nenhuma linha contém mais do que 80 letras.

Saída

Para cada instância imprima as N linhas traduzidas. As palavras que não estão no dicionário devem ser impressas como aparecem na entrada. Imprima uma linha em branco no final de cada tradução.

Nenhuma linha da saída contém mais do que 80 letras.

Exemplo de entrada	**Exemplo de saída**
1 4 3 galaxy cara tossiu kagayaku canalha do atsuki alto que yuushi util o galaxy o galaxy o kagayaku atsuki yuushi	o cara tossiu o cara tossiu o canalha do alto que util

D: Pie

`File: pie.[c|cpp|java|pas]`

My birthday is coming up and traditionally I'm serving pie. Not just one pie, no, I have a number N of them, of various tastes and of various sizes. F of my friends are coming to my party and each of them gets a piece of pie. This should be one piece of one pie, not several small pieces since that looks messy. This piece can be one whole pie though.

My friends are very annoying and if one of them gets a bigger piece than the others, they start complaining. Therefore all of them should get equally sized (but not necessarily equally shaped) pieces, even if this leads to some pie getting spoiled (which is better than spoiling the party). Of course, I want a piece of pie for myself too, and that piece should also be of the same size.

What is the largest possible piece size all of us can get? All the pies are cylindrical in shape and they all have the same height 1, but the radii of the pies can be different.

Input

One line with a positive integer: the number of test cases. Then for each test case:

1. One line with two integers N and F with $1 \leq N, F \leq 10000$: the number of pies and the number of friends.
2. One line with N integers r_i with $1 \leq r_i \leq 10000$: the radii of the pies.

Output

For each test case, output one line with the largest possible volume V such that me and my friends can all get a pie piece of size V. The answer should be given as a floating point number with exactly six digits in the fractional part.

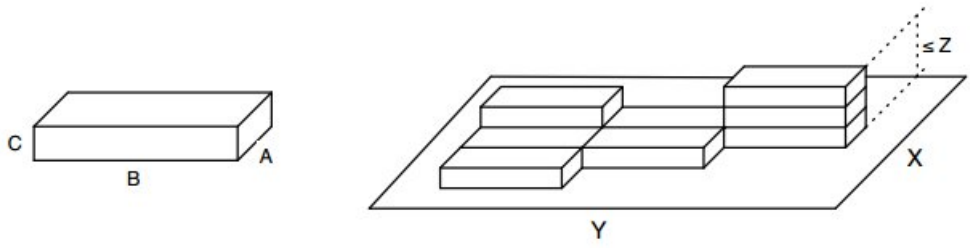
Sample Input	**Sample Output**
3 3 3 4 3 3 1 24 5 10 5 1 4 2 3 4 5 6 5 4 2	25.132741 3.141593 50.265482

E: Transporte

File: transporte.[c|cpp|java|pas]

A Betalândia é um país que apenas recentemente se abriu para o comércio exterior e está preparando agora sua primeira grande exportação. A Sociedade Betalandesa de Comércio (SBC) ficou encarregada de conduzir a exportação e determinou que, seguindo os padrões internacionais, a carga será transportada em contêineres, que são, por sua vez, colocados em grandes navios para o transporte internacional.

Todos os contêineres betalandeses são idênticos, medindo A metros de largura, B metros de comprimento e C metros de altura. Um navio porta-contêineres pode ser visto como um retângulo horizontal de X metros de largura e Y metros de comprimento, sobre o qual os contêineres são colocados. Nenhuma parte de contêiner pode ficar para fora do navio. Além disso, para possibilitar a travessia de pontes, a altura máxima da carga no navio não pode ultrapassar Z metros.



Devido a limitações do guindaste utilizado, os contêineres só podem ser carregados alinhados com o navio. Ou seja, os contêineres só podem ser colocados sobre o navio de tal forma que a largura e o comprimento do contêiner estejam paralelos à largura e ao comprimento do navio, respectivamente.

A SBC está com problemas para saber qual a quantidade máxima de contêineres que podem ser colocados no navio e pede sua ajuda. Sua tarefa, neste problema, é determinar quantos contêineres podem ser carregados no navio respeitando as restrições acima.

Entrada

A entrada contém vários casos de teste. Cada caso de teste consiste de duas linhas. A primeira linha contém três inteiros A, B e C que representam as dimensões dos contêineres, enquanto a segunda linha contém outros três inteiros X, Y e Z que representam as dimensões do navio.

A entrada termina com fim-de-arquivo (EOF). As restrições são:

- $1 \leq A, B, C, X, Y, Z \leq 10^6$
- É garantido que a maior resposta será menor ou igual a 10^6

Saída

Para cada caso de teste, seu programa deve imprimir apenas uma linha contendo um inteiro que indica a quantidade máxima de contêineres que o navio consegue transportar.

Exemplo de entrada	**Exemplo de saída**
1 1 1 1 1 1 1 2 5 9 6 11 1 2 12 6 9 10	1 54 0