5^o Treino para Alunos da UFPR

22 de Fevereiro de 2013

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

Organizadores:
Vinicius Kwiecien Ruoso, Ricardo Tavares de Oliveira e Flávio Zavan

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: The Device

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

Manao é responsável por gerenciar os equipamentos computacionais de sua universidade. Um certo dia, Manao recebeu a notícia de que a universidade comprou um novo equipamento, e de que ele ficará responsável por sua manutenção.

O equipamento comprado consiste em um dispositivo que recebe como entrada duas placas contendo sequências de *bits* de tamanho M cada e imprime como saída uma outra placa contendo uma sequência de tamanho M. O $i-$ésimo *bit* da sequência de saída é igual a um **e** (*AND*), um **ou** (*OR*) ou um **ou-exclusivo** (*XOR*) dos $i-$ésimos *bits* das sequências de entrada. A operação realizada em cada *bit* pode ser distinta.

Por exemplo, suponha que um dispositivo recebe como entrada placas com sequências de tamanho $M = 4$ e que o dispositivo corresponde, nesta ordem, às operações *(AND, OR, AND, XOR)*. Se alimentado com placas com as sequências $(1, 0, 1, 1)$ e $(0, 1, 1, 1)$, o dispositivo produzirá como saída uma placa com a sequência $(0, 1, 1, 0)$.

Manao sabe o valor de M, mas não sabe a configuração do dispositivo comprado, isto é, quais operações são realizadas em cada posição das entradas. Manao tem a sua disposição N placas contendo sequências de M *bits* cada. Quando o dispositivo chegar, Manao quer descobrir exatamente qual operação o dispositivo realiza em cada posição.

Para tal, Manao pode escolher qualquer par de placas distintas das N placas disponíveis, usá-las como entrada para o dispositivo e analisar sua saída. Manao pode fazer isto quantas vezes forem necessárias, com quantos pares de placas forem necessários. Sua tarefa é ajudar Manao e informar se o conjunto de N placas que Manao possui é suficiente para determinar exatamente a configuração do dispositivo, independente da configuração do dispositivo que chegar.

Entrada

A entrada consiste em um ou mais casos de teste. Cada caso começa com uma linha contendo N ($1 \leq N \leq 50$) e M ($1 \leq M \leq 50$). As próximas N linhas descrevem as placas que Manao tem. Cada placa é descrita por uma linha contendo M dígitos. É garantido que todos os digitos são iguais a 0 ou 1.

O último caso de teste é seguido por uma linha contendo dois zeros.

Saída

Para cada caso de teste, imprima *YES* se as placas de Manao são suficientes para determinar quais são as operações realizadas pelo dispositivo, e imprima *NO* caso contrário.

Exemplo de Entrada	Exemplo de Saída
3 3 010 011 000 4 1 1 0 1 0 1 5 11111 4 7 0110011 0101001 1111010 1010010 5 9 101001011 011011010 010110010 111010100 111111111 0 0	NO YES NO NO YES

B: 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

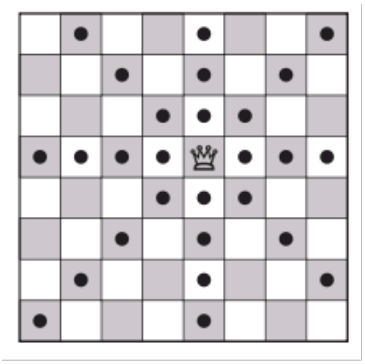
For each test case, output one line with the largest possible volume V such that me and my friends can all get a pie piece of size V. The answer should be given as a floating point number with exactly six digits in the fractional part.

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

C: Dama

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

O jogo de xadrez possui várias peças com movimentos curiosos: uma delas é a dama, que pode se mover qualquer quantidade de casas na mesma linha, na mesma coluna, ou em uma das duas diagonais, conforme exemplifica a figura abaixo:



O grande mestre de xadrez *Kary Gasparov* inventou um novo tipo de problema de xadrez: dada a posição de uma dama em um tabuleiro de xadrez vazio (ou seja, um tabuleiro 8×8, com 64 casas), de quantos movimentos, no mínimo, ela precisa para chegar em outra casa do tabuleiro?

Kary achou a solução para alguns desses problemas, mas teve dificuldade com outros, e por isso pediu que você escrevesse um programa que resolve esse tipo de problema.

Entrada

A entrada contem vários casos de teste. A primeira e única linha de cada caso de teste contém quatro inteiros X1, Y1, X2 e Y2 ($1 \leq$ X1, Y1, X2, Y2 ≤ 8). A dama começa na casa de coordenadas (X1, Y1), e a casa de destino é a casa de coordenadas (X2, Y2). No tabuleiro, as colunas são numeradas da esquerda para a direita de 1 a 8 e as linhas de cima para baixo também de 1 a 8. As coordenadas de uma casa na linha X e coluna Y são (X, Y).

O final da entrada é indicado por uma linha contendo quatro zeros.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma unica linha na saída, contendo um número inteiro, indicando o menor número de movimentos necessários para a dama chegar em sua casa de destino.

Exemplo de entrada	**Exemplo de saída**
4 4 6 2 3 5 3 5 5 5 4 3 0 0 0 0	1 0 2

D: Copa do Mundo

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

Uma Copa do Mundo de futebol de botões está sendo realizada com times de todo o mundo. A classificação é baseada no número de pontos ganhos pelos times, e a distribuição de pontos é feita da forma usual. Ou seja, quando um time ganha um jogo, ele recebe 3 pontos; se o jogo termina empatado, ambos os times recebem 1 ponto; e o perdedor não recebe nenhum ponto.

Dada a classificação atual dos times e o número de times participantes na Copa do Mundo, sua tarefa é de determinar quantos jogos terminaram empatados até o momento.

Input

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros T e N, indicando respectivamente o número de times participantes ($2 \leq T \leq 200$) e o número de partidas jogadas ($0 \leq N \leq 10000$). Cada uma das T linhas seguintes contém o nome de um time (uma cadeia de máximo 10 letras e dígitos), seguido de um espaço em branco, seguido do número de pontos que o time obteve até o momento. O final da entrada é indicado por T = 0.

Output

Para cada um dos casos de teste seu programa deve imprimir uma única linha contendo um número inteiro, representando a quantidade de jogos que terminaram empatados até o momento.

Sample Input	**Sample Output**
3 3 Brasil 3 Australia 3 Croacia 3 3 3 Brasil 5 Japao 1 Australia 1 0 0	0 2

E: Pascal Library

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

Pascal University, one of the oldest in the country, needs to renovate its Library Building, because after all these centuries the building started to show the effects of supporting the weight of the enormous amount of books it houses.

To help in the renovation, the Alumni Association of the University decided to organize a series of fund-raising dinners, for which all alumni were invited. These events proved to be a huge success and several were organized during the past year. (One of the reasons for the success of this initiative seems to be the fact that students that went through the Pascal system of education have fond memories of that time and would love to see a renovated Pascal Library.)

The organizers maintained a spreadsheet indicating which alumni participated in each dinner. Now they want your help to determine whether any alumnus or alumna took part in all of the dinners.

Input

The input contains several test cases. The first line of a test case contains two integers N and D indicating respectively the number of alumni and the number of dinners organized ($1 \leq N \leq 100$ and $1 \leq D \leq 500$). Alumni are identified by integers from 1 to N. Each of the next D lines describes the attendees of a dinner, and contains N integers X_i indicating if the alumnus/alumna i attended that dinner ($X_i = 1$) or not ($X_i = 0$). The end of input is indicated by $N = D = 0$.

Output

For each test case in the input your program must produce one line of output, containing either the word 'yes', in case there exists at least one alumnus/alumna that attended all dinners, or the word 'no' otherwise.

Alumna: a former female student of a particular school, college or university.
Alumnus: a former male student of a particular school, college or university.
Alumni: former students of either sex of a particular school, college or university.

Sample Input	**Sample Output**
3 3 1 1 1 0 1 1 1 1 1 7 2 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0	yes no