Problema A
Caixas Sem Limites

Nome do arquivo fonte: **caixas.c, caixas.cpp ou caixas.java**

Você se lembra do pintor Pares das Finais Mundiais da ACM do ano de 23451? Pares foi um dos inventores da monocromia, que significa que cada uma de suas pinturas tem uma única cor, mas em diferentes tons. Ele também acreditava na utilização de formas geométricas simples.

Há vários meses atrás, Pares pintava triângulos, em uma tela, de fora para dentro. Agora que os triângulos estão em baixa e os quadrados, em alta, suas mais recentes pinturas usam quadrados concêntricos, criados de dentro para fora! Pares começa a pintar sobre uma tela retangular dividida em uma grade quadrada perfeita. Ele seleciona um número de células individuais da grade para atuar como sementes centrais, e pinta estas com a cor mais escura. A partir de cada uma das sementes quadradas, Pares pinta um quadrado maior envolvente usando um tom mais claro, e repete com quadrados maiores envolventes, até que toda a tela seja coberta. Cada quadrado é exatamente uma célula a mais na grade e um tom mais claro em relação àquele que está envolvendo. Quando quadrados se sobrepõem, a célula da grade é sempre preenchida com a tonalidade mais escura.

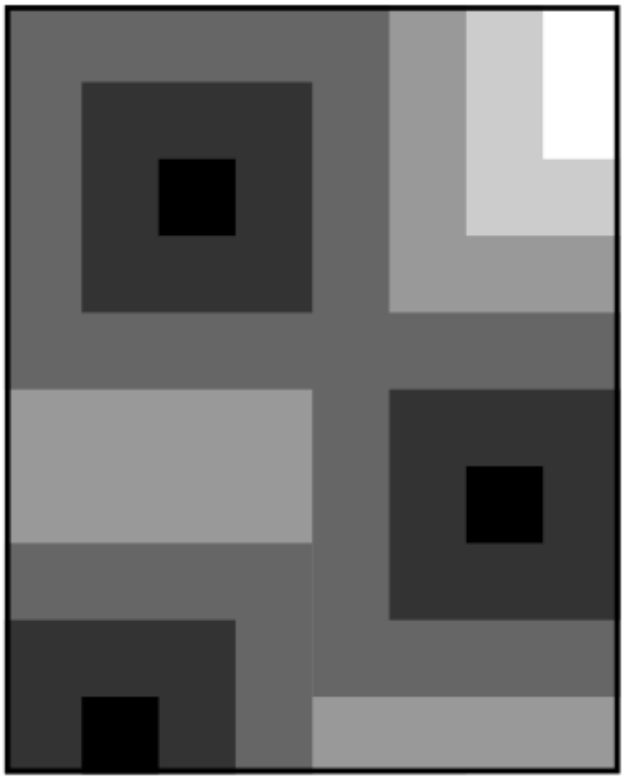

Figura 1: Exemplo de um dos mais recentes trabalhos de Pares, utilizando seis tonalidades de cor.

Depois de Pares decidir onde colocar os quadrados iniciais, a única parte difícil na criação dessas pinturas é decidir quantos tons diferentes de cor ele precisará. Para ajudar Pares, você deve escrever um programa que calcula o número de tons necessários para tal pintura, dado o tamanho da tela e os locais dos quadrados de sementes.

Entrada

O arquivo de entrada conterá múltiplos casos de teste. Cada caso de teste inicia com uma única linha contendo três números inteiros, m, n e s, separados por espaços. A tela contém exatamente m × n células de grade ($1 \leq m, n \leq 1000$), numeradas de 1, ..., m verticalmente e 1, …, n horizontalmente. Pares começa a pintura com s ($1 \leq s \leq 1000$) células de semente, descritas nas s linhas de texto seguintes, cada uma com dois inteiros, ri e ci ($1 \leq ri \leq m$, $1 \leq ci \leq n$), descrevendo a respectiva linha e coluna da grade de cada quadrado semente. Todos os quadrados sementes estão dentro dos limites da tela.

Uma linha em branco separa os casos de teste de entrada, como visto na amostra de entrada abaixo. Uma única linha com os números “0 0 0” marca o fim da entrada (não processe este caso).

Saída

Para cada caso de teste, seu programa deve imprimir um inteiro em uma única linha: o número de tons diferentes necessários à pintura descrita.

Exemplo:

Entrada	Saída
10 8 3 (Exemplo da Figura) 3 3 7 7 10 2 2 2 1 1 2 0 0 0	6 2

Problema B
Sequências

Nome do arquivo fonte: **sequencias.c, sequencias.cpp ou sequencias.java**

Descrição

Dorotéia, prima de Micalatéia, observou um fato interessante: uma sequência de números pode ser formada pela adição do quadrado de cada dígito ($452 = 4^2 + 5^2 + 2^2 = 45$) de um número para formar um novo número até que esta caia em um ciclo.

Por exemplo, partindo de 44 temos: 44 → 32 → 13 → 10 → 1 → 1, enquanto que, partindo de 85 temos a sequência: 85 → 89 → 145 → 42 → 20 → 4 → 16 → 37 → 58 → 89 (percebe-se uma repetição nestes dois casos: a primeira sequência convergirá para o valor 1 e a segunda sequência irá sempre repetir o ciclo iniciado pelo número 89).

Dorotéia, fascinada pelas sequências, pediu ajuda à Micalatéia, mas como ela estava muito ocupada com outras tarefas, decidiu pedir a ajuda aos maratonistas para desvendar os mistérios destas sequências. Dorotéia irá fornecer uma faixa de valores naturais [a,b] para números iniciais da sequência, para que você determine quantas sequências convergirão para o valor 1 e quantas sequências ficarão presas em um ciclo iniciado pelo valor 89.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é composto por dois inteiros a e b, que representam os extremos do intervalo que será avaliado, onde $0 \leq a \leq b \leq 10^7$, . O final da entrada é marcado por a = 0 e b = 0 (zeros são utilizados somente para marcar o final da entrada).

Saída

Para cada caso de teste seu programa deve imprimir em uma linha dois inteiros separados por um espaço, o primeiro sendo o número de sequências presas no valor 1 e o segundo o número de sequências presas no valor 89.

Exemplo:

Entrada	Saída
8 9 5 102 0 0	0 2 19 79

Problema C
Países em Guerra

Nome do arquivo fonte: **paises.c, paises.cpp ou paises.java**

Descrição

No ano 2050, após diversas tentativas da ONU de manter a paz no mundo, explode a terceira guerra mundial. Segredos industriais, comerciais e militares obrigaram todos os países a utilizar serviços de espionagem extremamente sofisticados, de forma que em cada cidade do mundo há ao menos um espião de cada país. Esses espiões precisam se comunicar com outros espiões, com informantes e mesmo com as suas centrais durante as suas ações. Infelizmente não existe uma forma segura de um espião se comunicar em um período de guerra, então as mensagens são sempre enviadas em código para que somente o destinatário consiga ler a mensagem e entender o seu significado.

Os espiões utilizam o único serviço que funciona no período de guerra, os correios. Cada cidade possui uma agência postal onde as cartas são enviadas. As cartas podem ser enviadas diretamente ao seu destino ou a outras agências postais, até que a carta chegue à agência postal da cidade de destino, se isso for possível.

Uma agência postal na cidade A pode enviar diretamente uma carta impressa para a agência postal da cidade B se houver um acordo de envio de cartas, que determina o tempo, em horas, que uma carta leva para chegar da cidade A à cidade B (e não necessariamente o contrário).Se não houver um acordo entre as agências A e B, a agência A pode tentar enviar a carta a quantas agências for necessário para que a carta chegue ao seu destino, se isso for possível.

Algumas agências são interligadas por meios eletrônicos de comunicação, como satélites e fibras ópticas. Antes da guerra, essas ligações atingiam todas as agências, fazendo com que uma carta fosse enviada de forma instantânea, mas durante o período de hostilidades cada país passou a controlar a comunicação eletrônica e uma agência somente pode enviar uma carta a outra agência por meio eletrônico (ou seja, instantaneamente) se ela estiver no mesmo país. Duas agências, A e B, estão no mesmo país se houver uma forma de uma carta impressa enviada de uma das agências ser entregue na outra agência.

O serviço de espionagem do seu país conseguiu obter o conteúdo de todos os acordos de envios de mensagens existentes no mundo e deseja descobrir o tempo mínimo para se enviar uma carta entre diversos pares de cidades. Você seria capaz de ajudá-lo?

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros separados por um espaço, N ($1 \leq N \leq 500$) e E ($0 \leq E \leq N^2$), indicando o número de cidades (numeradas de 1 a N) e de acordos de envio de mensagens, respectivamente. Seguem-se, então, E linhas, cada uma com três inteiros separados por espaços, X, Y e H ($1 \leq X, Y \leq N$, $1 \leq H \leq 1000$), indicando que existe um acordo para enviar uma carta impressa da cidade X à cidade Y , e que tal carta será entregue em H horas.

Em seguida, haverá uma linha com um inteiro K ($0 \leq K \leq 100$) que representa o número de

consultas. Finalmente, virão K linhas, cada uma representando uma consulta e contendo dois inteiros O e D ($1 \leq O, D \leq N$) separados por um espaço, os quais representam as cidades. Você deve determinar o tempo mínimo para se enviar uma carta da cidade O à cidade D.

Saída

Para cada caso de teste da entrada seu programa deve produzir K linhas na saída. A I-ésima linha deve conter um inteiro M , indicando o tempo mínimo, em horas, para se enviar uma carta na I-ésima consulta. Se não houver meio de comunicação entre as cidades da consulta, você deve imprimir ”Nao e possivel entregar a carta”(sem acentos).

Imprima uma linha em branco após cada caso de teste.

Exemplo:

Entrada	Saída
4 5 1 2 5 2 1 10 3 4 8 4 3 7 2 3 6 5 1 2 1 3 1 4 4 3 4 1 3 3 1 2 10 2 3 1 3 2 1 3 1 3 3 1 3 2 0 0	0 6 6 0 Nao e possivel entregar a carta 10 Nao e possivel entregar a carta 0

Problema D

Elementar meu caro Watson!

Nome do arquivo fonte: **wcw.c, wcw.cpp ou wcw.java**

Descrição

Watson, Crick e Wilkins receberam em 1962 o prêmio Nobel de Medicina especialmente pelo seu trabalho que resultou na descoberta da estrutura das moléculas do DNA e na sua importância na transmissão de informações entre as gerações de seres vivos. Watson e Crick publicaram na revista "Nature" em 1953 o artigo em que mostravam que a molécula de DNA apresentava uma estrutura de dupla hélice. O artigo assume enorme importância nos dias de hoje, especialmente depois dos vários avanços na área.

Muitas pesquisas têm sido feitas na área de Bioinformática ligadas à descoberta da seqüência de bases que compõem as moléculas de DNA dos vários seres vivos. Em especial, a estrutura destas moléculas tem sido usada para compor teorias de como os seres vivos evoluíram e quais têm ancestrais comuns. Acredita-se que os seres vivos presentes hoje no planeta podem descender de ancestrais comuns, sendo que as modificações nos seus respectivos DNAs ocorrem devido a fenômenos de mutação ocorridos durante a evolução. Muitos biólogos acreditam no princípio da parcimônia, que diz que o número destas mutações deve ser o mínimo possível, uma vez que a Natureza busca, de certa forma, o caminho "mais barato" para a modificação desejada.

Sua tarefa neste problema é auxiliar os pesquisadores na tarefa de determinar se duas seqüências de DNA podem ter um ancestral comum. Considere dadas duas seqüências (podemos imaginar como seqüências de números inteiros). O seu objetivo é determinar o menor número de trocas de elementos de uma das seqüências (os elementos não precisam estar em posições adjacentes na seqüência) que leva uma das seqüências na outra. Observe que podemos considerar uma das seqüências fixa (por exemplo, em ordem crescente), dessa forma buscamos o número mínimo de tais trocas que ordena a seqüência dada.

Entrada

A primeira linha da entrada possui um inteiro T que indica o número de instâncias. A primeira linha de cada instância possui um inteiro N ($1 \leq N \leq 10000$) indicando o número de inteiros na seqüência. A segunda linha contém uma permutação dos inteiros 1, 2, .., N separados por espaço.

Saída

Para cada instância imprima uma linha contendo o número mínimo de tais trocas que ordena a seqüência dada.

Exemplo:

Entrada	Saída
2 5 2 3 4 5 1 5 2 1 4 5 3	4 3

Problema E
Protegendo o Jardim

Nome do arquivo fonte: **jardim.c, jardim.cpp ou jardim.java**
Autor: **João Vítor Rocon Maia**

Descrição

No interior do Espírito Santo mora o alemão Hans e sua família. Hans é um grande fã de jardins e tem muito cuidado com o seu.

Pensando nisso Hans resolveu colocar um cachorro para tomar conta de seu jardim. O problema é que o cachorro do Hans é muito elétrico e não consegue ficar de guarda no jardim. A solução foi prender o seu cachorro a uma corda presa a um ponto fixo no jardim de forma a cobrir quase todo o jardim. Hans gostaria de proteger apenas alguns pontos específicos do jardim e para isso conta com sua ajuda.

Será fornecido um conjuntos de pontos (considere o sistema euclidiano) que são os pontos do jardim onde ele pretende proteger. Sua missão é encotrar o local do jardim onde a corda terá comprimento minimo de forma a proteger todos os pontos.

Entrada

Cada entrada começa com um interio N (3 <= N <= 1000) que corresponde a quantidade de pontos no jardim que Hans pretende proteger. No caso de N = 0, seu programa deve encerrar a execução.

Nas N linhas seguintes serão informados dois valores inteiros que correspondem a localização dos pontos do jardim (Eixo x e Eixo y nessa ordem).

Saída

Para cada entrada exibir na mesma linha as coordenadas X e Y de onde será fixada a corda e comprimento da corda.

Observação: sua resposta para cada um dos itens deve ter apenas 2 casa decimais de precisão.

Exemplo:

Entrada	Saída
5 1 1 2 4 5 0 2 3 2 3 3 0 0 3 0 0 3 4 2 0 -2 0 0 2 0 -2 0	3.35 1.88 2.51 1.50 1.50 2.12 0.00 0.00 2.00

Problema F
Rotações e Reflexões

Nome do arquivo fonte: **rotacao.c, rotacao.cpp ou rotacao.java**

Descrição

Muitos jogos, brincadeiras e quebra-cabeças dependem em determinar se dois padrões sobre uma grade retangular são os "mesmos" ou não. Por exemplo, no clássico problema das 8 rainhas, temos 96 maneiras diferentes de arranjar estas 8 rainhas seguramente (sem que nenhuma se ataque mutuamente) sobre um tabuleiro de xadrez. Contudo, pode ser mostrado que estas 96 soluções consistem de rotações e/ou reflexões a partir de apenas 12 padrões básicos.

Escreva um programa que leia pares de padrões e determine se existe uma simples transformação que converterá um no outro. Devido os padrões simétricos conhecidos, há muitos relacionamentos um com o outro, e as transformações que devem ser verificadas seguem numa específica ordem/seqüência. As possíveis transformações (em ordem) são dadas por:

Preservar	Os padrões são identicos
Rotação de 90 graus	O padrão foi rotacionado de 90 graus no sentido horário
Rotação de 180 graus	O padrão foi rotacionado de 180 graus no sentido horário
Rotação de 270 graus	O padrão foi rotacionado de 270 graus no sentido horário
Reflexão	O padrão foi refletido sobre o eixo horizontal (eftivamente um espelho preso na parte superior do padrão)
Combinação	Uma reflexão (acima descrito), seguido por uma das rotações acima)
Impróprio	Os padrões não casam sob nenhuma das transforamações acima

Entrada

A entrada consistirá de uma série de pares de padrões. Cada conjunto consistirá de uma linha contendo um simples inteiro `N` (2 <= N <= 10) o qual fornece o tamanho do padrão, seguido pelas `N` linhas do padrão. Cada linha consistirá de `N` pontos (`'.'`) ou `'x'`s (especificando uma linha do padrão original), em seguida um espaço, e um outro conjunto de `N` pontos (`'.'`) e `'x'`s (especificando uma linha do padrão transformado). Isto é, o início do segundo padrão a ser comparado. O arquivo terminará por uma linha consistindo de um simples zero (`0`).

Saída

A saída consistirá de uma série de linhas, uma para cada par de padrão da entrada. Cada linha consistirá em uma das seguintes mensagens como saída: ‘`Preserved`’, ‘`Rotated through m degrees`’ (onde m é umvalor de 90, 180 ou 270), ‘`Reflected`’, ‘`Reflected and rotated through m degrees`’, ‘`Improper`’.

Exemplo:

Entrada	Saída
5 x...x ....x .x... ...x. ...x. .x... ..x.x ..x.. ....x xx..x 2 x. xx x. xx 4 ..x. ...x xx.. .... xx.. ...x ..x. 4 .x.. ..x. .x.x x... .... ..xx ..x. 0	Rotated through 90 degrees Improper Reflected Reflected and rotated through 270 degrees

Problema G
Corrida dos Marrecos

Nome do arquivo fonte: **corrida.c, corrida.cpp ou corrida.java**

Descrição

Pirabeiraba é um distrito de Joinville, onde colonizadores alemães se instalaram no início do século XX. Anualmente há a festa do aipim, tubérculo conhecido como macaxeira no nordeste do Brasil. Para acompanhar o aipim, nada como um prato típico germânico: o marreco recheado! Para os entendidos de culinária, há uma magia nesta combinação: marreco com aipim. Contudo, para matar o marreco, você deve capturá-lo quando este estiver com o sangue bem quente. Para isto, o marreco deve estar cansado. Dizem que seu sangue quente é sinônimo de fertilidade, para não dizer: afrodisíaco! Mas isto é outra história.

Nesta brincadeira de correr atrás do marreco, surgiu a idéia de cansá-los com uma corrida entre eles. O espaço físico da Sociedade Rio da Prata é limitado, assim, construíram apenas 3 raias para se realizar estas corridas. As corridas são feitas em grupos de 2 e 3 marrecos. Os primeiros colocados destes grupos são novamente divididos em grupos de 2 e 3 para uma nova rodada. Isto acontece até que só reste o marreco campeão, que, como prêmio foge (por ora) da panela. Todos os marrecos sobreviventes devem correr na rodada, isto é, se não for possível dividir todos os marrecos em grupos de 3, alguns grupos de 2 devem ser formados, mas de forma a minimizar o número de corridas.

Exemplos são vistos na figura abaixo:

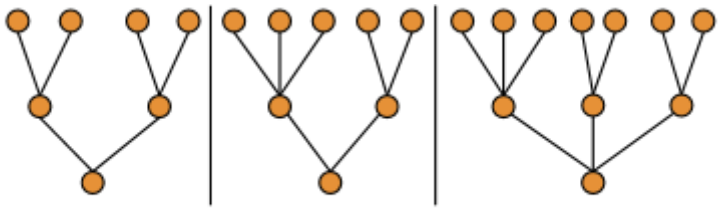

Figura 1: Exemplos: competição com 4, 5 e 7 marrecos.

Os marrecos perdedores, por sua vez, serão os primeiros a irem para panela. Você foi convidado para comer marreco com aipim, mas, em troca, deve escrever um programa que calcule o número de corridas realizadas para se determinar o marreco campeão.

Entrada

A entrada do programa é composta por vários casos de teste. Cada caso de teste é composto por uma linha contendo um número n (número de marrecos), tal que 1 <= n <= 100000, e n = 0 é utilizado unicamente para marcar o término das entradas, sendo que este deve ser desconsiderado.

Saída

O seu programa deve imprimir na saída padrão uma linha por caso de teste, contendo o número de corridas necessárias para escolher o marreco campeão.

Exemplo:

Entrada	Saída
3 4 5 6 7 0	1 3 3 3 4

Problema H
O Equilíbrio do Mundo

Nome do arquivo fonte: **equilibrio.c, equilibrio.cpp ou equilibrio.java**

Descrição

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 sequência/frase dada ao programa pode ter dois tipos de delimitadores, os parênteses (arrendodados) ("( )") e colchetes (quadrados) ("[ ]"). Uma sequência 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 dos 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 conjundo de dados é uma sequência de 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 ultimo 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:

Entrada	Saída
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)]. ([(([([ ]) ( ) (( ))]))]). . .	yes yes no no no yes yes

Problema I
Maior Quantidade de Vendas

Nome do arquivo fonte: **vendas.c, vendas.cpp ou vendas.java**
Autor: **João Vítor Rocon Maia**

Descrição

Você faz parte da equipe de desenvolvimento do Sucatas.com.br que é um e-commerce focado em vender produtos usados. Seu chefe resolveu criar uma nova área do sistema administrativo e conta com o conhecimento de sua equipe para criar essa nova funcionalidade.

É uma verdade que aplicações com o passar dos tempos crescem em termos de volume de dados e, quanto maior for o tempo e maior a quantidade de usuários, estes numeros podem chegar a valores assustadores.

Na última reunião foi definido que na aplicação da sua empresa será criado um widget novo onde poderá ser definido uma data de inicio e fim e o resultado será o valor de produtos vendidos no dia que teve mais vendas.

Sabendo do problema do volume de dados, sua equipe estudou a implementação de um algoritmo eficiente para resolver esse problema. Como forma de prova de conceito, sua equipe decidiu implementar uma abstração do problema usando uma lista de inteiros simples.

Sua missão é escrever esse algoritmo de prova de conceito.

Entrada

A primeira linha de cada caso de teste é composta por um valor N (2 <= N <= 50000) onde N é a quantidade de dias. No caso de N = 0, seu programa deve encerrar a execução. Na linha seguinte serão dados N valores de vendas separados por espaço.

Na terceira linha será dado um valor M (1 <= M <= 100) onde M é a quantidade de consultas a serem realizadas na sua coleção de dados sobre a quantidade de vendas. As M linhas seguintes são compostas por dois inteiros separados por espaço, o primeiro valor será sempre menor ou igual ao segundo.

Observação: considere que a lista contendo os N dias possui o indíce começando por 0 (Zero).

Saída

Para cada entrada, deve existir uma linha indicando o número do teste. Após isso, cada linha seguinte deve retornar o valor máximo das consultas realizadas.

Os testes devem ser separados por uma linha em branco. No último caso de teste, a linha em branco deve ser omitida.

Exemplo:

Entrada	Saída
10 1 10 2 9 3 8 4 7 5 6 5 0 9 1 3 3 8 7 8 2 5 2 1 3 1 0 1 0	Entrada #1: 10 10 9 7 9 Entrada #2: 3

Problema J
Los Buses de Cartagena

Nome do arquivo fonte: **marques.c, marques.cpp ou marques.java**

Descrição

Gabriel Garcia Marques é um escritor colombiano autor de histórias fantásticas como “Cién años de soledade”, “El amor en los tiempos del cólera” e “Memoria de mis putas tristes”. Suas histórias se caracterizam pelo uso do que ficou conhecido como “realismo mágico”, em que situações reais são explicadas com elementos mágicos. Apesar de seus trabalhos serem considerados muito ricos e até cenográficos, livros baseados em suas obras não têm merecido sucesso de público ou de crítica. O mais recente exemplo foi a filmagem em 2007 de “Love in the Time of Cholera”.

Uma de suas obras menos conhecidas é “Los buses de Cartagena”, que descreve a história de uma pequena companhia de ônibus da cidade colombiana que, principalmente devido aos problemas de quebra dos ônibus por excesso de carga, pretendia reduzir o número de passageiros transportados em cada viagem de Cartagena a Medellin para um mesmo número fixo. Ao mesmo tempo, a companhia queria continuar atendendo a todos os pedidos de forma satisfatória. Cada ônibus possui um horário de partida, e cada passageiro dispõe de uma lista de horários nos quais gostaria de viajar. Os passageiros desejam apenas ir para Medellin, ou seja, nenhum passageiro pretende viajar duas vezes no mesmo dia.

Sua tarefa é determinar o número mínimo de passageiros que devem ser transportados em cada viagem respeitando a restrição de que todos os passageiros devem ser atendidos.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias.

A primeira linha de cada instância contém dois inteiros N e M (1 <= N, M <= 100). Cada uma das M linhas seguintes possui o horário de partida de um dos ônibus. O horário está no formato hh:mm (00 <= hh <= 23, 00 <= mm <= 59 e hh e mm possuem dois dígitos). Cada uma das N linhas seguintes contém a lista de horários em que cada passageiro pode viajar. A lista dos horários está no seguinte formato: um inteiro K (1 <= K <= M) seguido de K horários, também no formato hh:mm, separados por um espaço em branco.

Saída

Para cada instância imprima uma linha contendo o número mínimo de passageiros que devem ser transportados.

Exemplo:

Entrada	Saída
3 3 2 00:10 11:30 1 00:10 2 00:10 11:30 2 11:30 00:10 3 3 23:50 23:50 23:51 2 23:51 23:50 1 23:50 1 23:50 4 2 10:00 12:01 1 12:01 1 12:01 1 12:01 1 12:01	2 1 4