Problema A
Pontos cocirculares

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

Você provavelmente sabe o que é um conjunto de pontos colineares: é um conjunto de pontos tal que existe uma linha reta que passa através de todos os pontos. Um conjunto de pontos cocirculares é definido da mesma forma, mas ao invés de uma linha reta, nós queremos saber se existe um círculo tal que todos os pontos do conjunto encontram-se sobre seu perímetro.

A International Collinear Points Center(ICPC) designou a você a seguinte tarefa: dado um conjunto de pontos, calcule o tamanho do maior subconjunto de pontos cocirculares.

Entrada

Cada caso de teste se estende por várias linhas. A primeira linha contém um inteiro **N** representando o número de pontos no conjunto ($1 \leq N \leq 100$). Cada uma das próximas **N** linhas contém dois inteiros X e Y representando as coordenadas de um ponto de conjunto ($-10^4 \leq X,Y \leq 10^4$). Em cada caso de teste, não haverá dois pontos com mesma localização.
O último caso de teste é seguido por uma linha contendo apenas um zero.

Saída

Para cada caso de teste, imprima uma única linha com um único inteiro representando o número de pontos em um dos maiores subconjuntos da entrada que são cocirculares.

Exemplo:

Entrada	Saída
7 -10 0 0 -10 10 0 0 10 -20 10 -10 20 -2 4 4 -10000 10000 10000 10000 10000 -10000 -10000 -9999 3	5 3 2

-1 0 0 0 1 0 0	

Problema B
Robô colecionador

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

Um dos esportes favoritos na Robolândia é o Rali de Robôs. Este rali é praticado em uma arena retangular gigante de N linhas por M colunas de células quadradas. Algumas das células estão vazias, algumas contêm figurinhas da Copa (muito apreciadas pelas inteligências artificiais da Robolândia) e algumas são ocupadas por pilastras que sustentam o teto da arena. Em seu percurso os robôs podem ocupar qualquer célula da arena, exceto as que contém pilastras, que bloqueiam o seu movimento. O percurso do robô na arena durante o rali é determinado por uma sequência de instruções. Cada instrução é representada por um dos seguintes caracteres: 'D', 'E' e 'F', significando, respectivamente, “gire 90 graus para direita”, “gire 90 graus para esquerda” e “ande uma célula para frente”. O robô começa o rali em uma posição inicial na arena e segue fielmente a sequência de instruções dada (afinal, eles são robôs!). Sempre que o robô ocupa uma célula que contém uma figurinha da Copa ele a coleta. As figurinhas da Copa não são repostas, ou seja, cada figurinha pode ser coletada uma unica vez. Quando um robô tenta andar para uma célula onde existe uma pilastra ele patina, permanecendo na célula onde estava, com a mesma orientação. O mesmo também acontece quando um robô tenta sair da arena.

Dados o mapa da arena, descrevendo a posição de pilastras e figurinhas, e a sequência de instruções de um robô, você deve escrever um programa para determinar o número de figurinhas coletadas pelo robô.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três números inteiros **N**, **M** e **S** ($1 \leq \mathbf{N},\mathbf{M} \leq 100$, $1 \leq \mathbf{S} \leq 5 \times 10^4$), separados por espaços em branco, indicando respectivamente o número de linhas e o número de colunas da arena e o número de instruções para o robô. Cada uma das **N** linhas seguintes da entrada descreve uma linha de células da arena e contém uma cadeia com **M** caracteres. A primeira linha que aparece na descrição da arena é a que está mais ao Norte; a primeira coluna que aparece na descrição de uma linha de células da arena é a que está mais a Oeste.

Cada célula da arena pode conter um dos seguintes caracteres:

- '.' - célula normal;
- '*' - célula que contém uma figurinha dacopa;
- '#' - célula que contém uma pilastra;
- 'N','S','L','O' – célula onde o robô inicia o percurso (única na arena). A letra representa a orientação inicial do robô (Norte, Sul, Leste e Oeste, respectivamente).

A ultima linha da entrada contém uma sequência de **S** caracteres dentre ‘D’, ‘E’ e ‘F’, representando as instruções do robô.
O último caso de teste é seguido por uma linha que contém apenas três números zero separados por um espaço em branco.

Saída

Para cada rali descrito na entrada seu programa deve imprimir uma única linha contendo um único inteiro, indicando o número de figurinhas que o robô colecionou durante o rali.

Exemplo:

Entrada	Saída
3 3 2 *** *N* *** DE 4 4 5 ...# *#O. *.*. *.#. FFEFF 10 10 20 ....*..... ....*.. .....*.... ..*.#..... ...#N.*..* ...*..... .... FDFFFFFFEEFFFFFFEFDF 0 0 0	0 3 1

Problema C
Eletricidade

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

Martin e Isa pararam de jogar jogos loucos e finalmente se casaram. Ótimas notícias! Eles estão vivendo uma nova vida de felicidade para ambos e, também, estão se mudando para uma nova casa em um lugar remoto, comprado com a maior parte de suas economias.

A vida é diferente nesse novo lugar. Particularmente, a energia elétrica é muito cara, e eles querem manter tudo sob controle. Por isso Martin propôs que mantivessem um histórico diário de quanta eletricidade foi consumida na casa. Eles têm um marcador de eletricidade, que mostra um número com a quantidade de KWh (kilowatts-hora) que foi consumida desde sua chegada.

No começo de cada dia eles consultam o marcador de eletricidade, e anotam o consumo. Alguns dias Martin faz isso, em outros é a Isa quem faz. Desse jeito, eles conseguirão observar as diferenças de consumo entre dias consecutivos e saber quanto foi gasto.

Mas alguns dias eles simplesmente esqueceram de anotar, então, depois de muito tempo, o histórico está incompleto. Eles têm uma lista de datas e consumos, mas nem todas datas são consecutivas. Eles só querem levar em conta os dias para os quais o consumo pode ser determinado precisamente, e precisam de ajuda.

Entrada

A entrada contém diversos casos de teste. A primeira linha de um caso de teste contém um inteiro N indicando o número de medições que eles fizeram ($2 <= N <= 10^3$). Cada uma das N linhas seguintes contém quatro inteiros D, M, Y e C, separados por espaços, indicando respectivamente o dia ($1 <= D <= 31$), mês ($1 <= M <= 12$), ano ($1900 <= Y <= 2100$), e consumo ($0 <= C <= 10^6$) lidos no início de cada dia. Essas N linhas são ordenadas em ordem crescente pela data e podem incluir anos bissextos. A seqüência de consumos é estritamente crescente (isto é, duas leituras sempre têm valores diferentes). Você pode assumir que D, M e Y representam datas válidas.

Regras para determinação se um ano é bissexto:

1. De 4 em 4 anos é ano bissexto.
2. De 100 em 100 anos não é ano bissexto.
3. De 400 em 400 anos é ano bissexto.
4. Prevalecem as últimas regras sobre as primeiras.

O final da entrada é indicado por uma linha contendo apenas um zero.

Saída

Para cada caso de teste na entrada, seu programa deve imprimir uma única linha contendo dois inteiros separados por um único espaço: o número de dias para os quais o consumo pode ser

determinado precisamente e o consumo desses dias.

Exemplo:

Entrada	Saída
5 9 9 1979 440 29 10 1979 458 30 10 1979 480 1 11 1979 483 2 11 1979 483 3 5 5 2000 6780 6 5 2001 7795 7 5 2002 8201 8 28 2 1978 112 1 3 1978 113 28 2 1980 220 1 3 1980 221 5 11 1980 500 14 11 2008 600 15 11 2008 790 16 12 2008 810 0	2 15 0 0 2 191

Problema D
Desvio de Rota
Nome do arquivo fonte: **desvio.c, desvio.cpp ou desvio.java**

O sistema rodoviário de um país interliga todas as suas **N** cidades de modo que, a partir de uma cidade qualquer, é possível chegar a cada uma das outras cidades trafegando pelas estradas existentes. Cada estrada liga duas cidades distintas, tem mão dupla e um único posto de pedágio (o pedágio é pago nos dois sentidos de tráfego). As estradas não se intersectam a não ser nas cidades. Nenhum par de cidades é interligado por duas ou mais estradas.

A Transportadora Dias oferece um serviço de transporte de encomendas entre as cidades. Cada encomenda deve ser levada de uma cidade A para uma outra cidade B. A direção da Transportadora Dias define, para cada encomenda, uma rota de serviço, composta por **C** cidades e **C**−1 estradas: a primeira cidade da rota de serviço é a origem da encomenda, a última o destino da encomenda. A rota de serviço não passa duas vezes pela mesma cidade, e o veículo escolhido para fazer o transporte de uma encomenda pode trafegar apenas pela rota de serviço definida.

Certo dia, no entanto, o veículo que executava uma entrega quebrou e precisou ser levado para conserto em uma cidade que não está entre as cidades de sua rota de serviço. A direção da Transportadora Dias quer saber qual é o menor custo total, em termos de pedágio, para que o veículo entregue a encomenda na cidade destino, a partir da cidade em que foi consertado, mas com uma restrição adicional: se em algum momento o veículo passar por uma das cidades que compõem a sua rota de serviço, ele deve voltar a obedecer a rota de serviço.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros **N**, **M**, **C** e **K** ($4 \le N \le 250$, $3 \le M \le N \times (N-1)/2$, $2 \le C \le N-1$ e $C \le K \le N-1$), representando, respectivamente, o número de cidades do país, o número de estradas, o número de cidades na rota de serviço e a cidade em que o veículo foi consertado. As cidades são identificadas por inteiros de 0 a **N**−1. A rota de serviço é 0, 1, ... , **C**−1, ou seja, a origem é 0, de 0 passa para 1, de 1 para 2 e assim por diante, até o destino **C**−1.

As **M** linhas seguintes descrevem o sistema rodoviário do país. Cada uma dessas linhas descreve uma estrada e contém três inteiros **U**, **V** e **P** ($0 \le U, V \le N-1$, $U \ne V$, $0 \le P \le 250$), indicando que há uma estrada interligando as cidades**U** e **V** com custo de pedágio **P**. O último caso de teste é seguido por uma linha contendo quatro zeros separados por espaço em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha, contendo um único inteiro **T**, o custo total mínimo necessário, em termos de pedágio, para que o veículo chegue ao destino.

Exemplo:

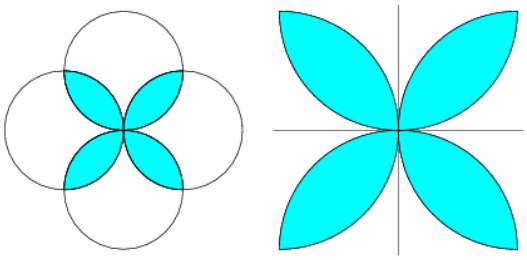
Entrada	Saída
4 6 3 3 0 1 10 1 2 10 0 2 1 3 0 1 3 1 10 3 2 10 6 7 2 5 5 2 1 2 1 10 1 0 1 3 0 2 3 4 2 3 5 3 5 4 2 5 5 2 4 0 1 1 1 2 2 2 3 3 3 4 4 4 0 5 0 0 0 0	10 6 6

Problema E
Trevo perfeito

Nome do arquivo fonte: **trevo.c, trevo.cpp ou trevo.java**
Autor: Maycon Maia Vitali (LPRM)

Um trevo de quatro folhas é uma folha de trevo que apresenta quatro em vez dos normais três folíolos comuns na maioria das espécies do gênero *Trifolium* a que pertencem os trevos. Com origem nas antigas tradições dos povos celta, acredita-se que encontrar um trevo-de-quatro-folhas é um sinal de boa sorte.

Uma subcategoria de um trevo-de-quatro-folhas é o trevo perfeito, extremamente raro e encontrado somente em meio à flora da Universidade Federal do Espírito Santo (UFES). Um trevo perfeito tem como características a possibilidade desenhá-lo utilizando 4 circunferências idênticas e organizadas como visto na figura abaixo:



Estudiosos dizem que um chá feito desse trevo cura a doença causada pelo vírus *Fanfarras a Limine ab Aeterno*, comumente encontrada em meio aos que frequentam as proximidades do CT-VII. Com isso, o professor de Botânica IV do curso de Biologia da UFES, Joseph Bisnacan, propôs para seus alunos que encontrem trevos perfeitos no campus, e em compensação ganharão a não aplicação de provas e o término do semestre letivo 2 meses antes do previsto no calendário.

Para a fabricação do tal chá “milagroso”, é necessário ter uma quantidade específica de trevos perfeitos. Para isso deseja-se saber a área da superfície das folhas de um determinado trevo, tendo somente o raio da circunferência que o forma.

Entrada

A entrada consiste de vários casos de testes. Cada caso de teste é representado por um único número real R ($0 < R \leq 10^6$), que representa o raio da circunferência que forma o trevo perfeito. A entrada termina com o final do arquivo. Seu programa deve ler os dados da entrada padrão.

Saída

Para cada caso de teste seu programa deve imprimir uma única linha, contendo um número real,

escrito com precisão de 3 casa decimais, indicando a área da superfície do trevo perfeito. O resultado de seu programa deve ser escrito na saída padrão.

Exemplo:

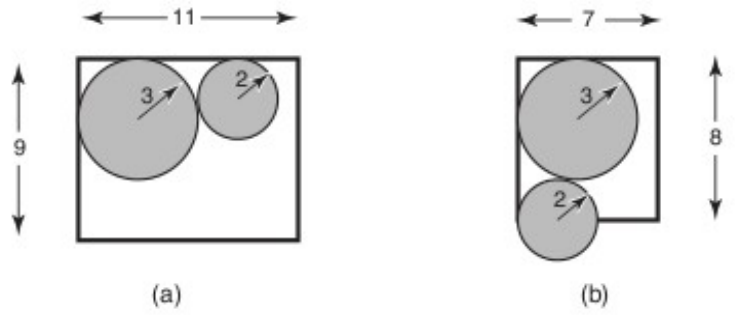
Entrada	Saída
2 5 11.5	9.133 57.080 301.951

Problema F
Elevador

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

A FCC (Fábrica de Cilindros de Carbono) fabrica vários tipos de cilindros de carbono. A FCC está instalada no décimo andar de um prédio, e utiliza os vários elevadores do prédio para transportar os cilindros. Por questão de segurança, os cilindros devem ser transportados na posição vertical; como são pesados, no máximo dois cilindros podem ser transportados em uma única viagem de elevador. Os elevadores têm formato de paralelepípedo e sempre têm altura maior que a altura dos cilindros.

Para minimizar o número de viagens de elevador para transportar os cilindros, a FCC quer, sempre que possível, colocar dois cilindros no elevador. A figura abaixo ilustra, esquematicamente (vista superior), um caso em que isto é possível (a), e um caso em que isto não é possível (b):



Como existe uma quantidade muito grande de elevadores e de tipos de cilindros, a FCC quer que você escreva um programa que, dadas as dimensões do elevador e dos dois cilindros, determine se é possível colocar os dois cilindros no elevador.

Entrada

A entrada contém vários casos de teste. A primeira e única linha de cada caso de teste contém quatro números inteiros **L**, **C**,**R1** e **R2**, separados por espaços em branco, indicando respectivamente a largura do elevador ($1 \leq \mathbf{L} \leq 100$), o comprimento do elevador ($1 \leq \mathbf{C} \leq 100$), e os raios dos cilindros ($1 \leq \mathbf{R1}, \mathbf{R2} \leq 100$).

O último caso de teste é seguido por uma linha que contém quatro zeros separados por espaços em branco

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha com um único caractere: ‘S’

se for possível colocar os dois cilindros no elevador e ‘N’ caso contrário.

Exemplo:

Entrada	Saída
11 9 2 3 7 8 3 2 10 15 3 7 8 9 3 2 0 0 0 0	S N N S

Problema G
Olimpíadas

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

Tumbólia é um pequeno país ao leste da América do Sul (ou ao sul da América do Leste) que irá participar dos Jogos Olímpicos pela primeira vez na sua história. Apesar de sua delegação ser muito pequena comparada ao total de atletas que estarão em Pequim (as estimativas oficiais são de mais de dez mil atletas), a participação será fundamental para a imagem e para o turismo de Tumbólia.

Após selecionar os atletas, o Comitê Olímpico Tumboliano (COT) precisa comprar as passagens para eles. A fim de economizar dinheiro, o COT decidiu comprar apenas passagens da Air Rock. No entanto, muitas das passagens da Air Rock já foram vendidas, uma vez que muitos tumbolianos desejam assistir aos Jogos. Sendo assim, o COT deverá comprar passagens de acordo com os assentos vagos em cada vôo.

Todos os vôos da Air Rock partem diariamente antes do meio-dia e chegam após o meio- dia; por isso, um atleta pode tomar apenas um avião por dia. A Air Rock providenciou uma lista contendo todos os vôos operados por ela e o número de assentos vagos em cada um (curiosamente, o número de assentos livres em um mesmo trecho é igual todos os dias).

O COT verificou que realmente é possível ir de Tumbólia para Pequim usando apenas vôos da Air Rock mas, mesmo assim, o COT está tendo dificuldades para planejar a viagem de seus atletas. Por isso, o COT pediu para você escrever um programa que, dada a lista de vôos da Air Rock, determina a menor quantidade de dias necessária para que todos os atletas cheguem em Pequim.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém três inteiros N, M e A indicando respectivamente a quantidade de aeroportos em que a Air Rock opera ($2 \leq N \leq 50$), a quantidade de vôos em que há assentos vagos ($1 \leq M \leq 2.450$) e quantos atletas a delegação tumboliana tem ($1 \leq A \leq 50$).

Cada uma das M linhas seguintes contém uma descrição de vôo com três inteiros O, D e S que indicam respectivamente o aeroporto de origem ($1 \leq O \leq N$), o aeroporto de destino ($1 \leq D \leq N$ e $O \neq D$) e a quantidade de assentos vagos naquele vôo ($1 \leq S \leq 50$). Os aeroportos são numerados de 1 a N; o Aeroporto Internacional de Tumbólia é o aeroporto1, e o Aeroporto Internacional de Pequim é o aeroporto N.

A existência de um vôo de A para B não implica a existência de um vôo de B para A (mas sempre há no máximo um vôo de um aeroporto para outro em cada direção).

O final da entrada é indicado por N = M = A = 0. A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída contendo um inteiro, indicando a quantidade mínima de dias necessária para que todos os atletas tumbolianos cheguem em Pequim (alguns atletas podem chegar depois de outros, e eles não precisam chegar na mesma ordem em que partiram).

A saída deve ser escrita na saída padrão.

Exemplo:

Entrada	Saída
3 3 3 1 2 2 2 3 2 1 3 1 3 3 5 1 2 1 2 3 5 3 1 4 4 4 4 1 4 1 1 2 1 2 3 1 3 4 1 0 0 0	2 6 3

Problema H
Apagando e ganhando

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

Juliano é fã do programa de auditório Apagando e Ganhando, um programa no qual os participantes são selecionados atráves de um sorteio e recebem prêmios em dinheiro por participarem.

No programa, o apresentador escreve um número de N dígitos em uma lousa. O participante então deve apagar exatamente Ddígitos do número que está na lousa; o número formado pelos dígitos que restaram é então o prêmio do participante.

Juliano finalmente foi selecionado para participar do programa, e pediu que você escrevesse um programa que, dados o número que o apresentador escreveu na lousa, e quantos dígitos Juliano tem que apagar, determina o valor do maior prêmio que Juliano pode ganhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros N e D (1 <= D < N <= 10^5), indicando a quantidade de dígitos do número que o apresentador escreveu na lousa e quantos dígitos devem ser apagados. A linha seguinte contém o número escrito pelo apresentador, que não contém zeros à esquerda.

O final da entrada é indicado por uma linha que contém apenas dois zeros, separados por um espaço em branco.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha na saída, contendo o maior prêmio que Juliano pode ganhar.

Exemplo:

Entrada	Saída
4 2 3759 6 3 123123 7 4 1000000 0 0	79 323 100

Problema I
Bases

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

O que você consegue se multiplicar 6 por 9? A resposta, é claro, é 42, mas somente se você fizer os cálculos na base 13.
Dado um inteiro B >= 2, o sistema de numeração na base B é a maneira de escrever inteiros usando somente dígitos entre 0 e B - 1, inclusive. Em um número escrito na base B, o dígito mais à direita tem seu valor multiplicado por 1, o segundo mais à direita tem seu valor multiplicado por B, o terceiro mais à direita tem seu valor multiplicado por B^2, e assim por diante.

Algumas equações são verdadeiras ou falsas dependendo da base em que são consideradas. A equação 2 + 2 = 4, por exemplo, é verdadeira para qualquer B >= 5 - ela não vale para a base 4, por exemplo, visto que não existe dígito '4' na base 4. Por outro lado, uma equação como 2 + 2 = 5 nunca é verdadeira.

Escreva um programa que, dada uma equação, determine em quais bases ela é verdadeira.

Entrada

Cada linha da entrada contém um caso de teste; cada caso de teste é uma equação da forma "EXPR=EXPR", onde ambos "EXPR" são expressões aritméticas com no máximo 17 caracteres.

Todas expressões são válidas e contém apenas os caracteres '+', '*' e os dígitos entre '0' e '9'. Nenhuma expressão contém sinais de mais no começo da equação e nenhum número tem zeros à esquerda

O final da entrada é indicado por uma linha contendo apenas "=".

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha de saída, indicando para quais bases a equação dada é válida

Se a expressão for verdadeira para infinitas bases, imprima "B+", onde B é a primeira base para a qual a equação é válida.

Se a expressão for válida apenas para um conjunto finito de bases, imprima elas em ordem crescente, separadas por espaço.

Se a expressão não for verdadeira em nenhuma base, imprima o caractere '*'.

Exemplo:

Entrada	Saída
6*9=42 10000+3*5*334=3*5000+10+0 2+2=3 2+2=4 0*0=0 =	13 6 10 * 5+ 2+

Problema J
Senha

Nome do arquivo fonte: **senha.c, senha.cpp ou senha.java**
Autor: Maycon Maia Vitali (LPRM)

Maycon é um aluno de Mestrado no Laboratório de Pesquisas da Roberta e o Magnos (LPRM) e [quase] sempre um dos primeiros a chegar no prédio do CT-IX, abrindo o cadeado de combinação de 4 (quatro) dígitos que se encontra no portão principal. Devido esse fato ter ocorrido diversas vezes nos últimos anos, Maycon percebeu que criou uma capacidade extra humana de chegar na combinação correta do cadeado dada uma combinação qualquer. Com isso, ele resolveu criar o Instituto de Computação de Práticas de Cadeados (ICPC), que promove anualmente um torneio de abertura de cadeados de combinação.

As regras do torneio são simples. O participante só pode movimentar de um número para outro por vez, ou seja, para sair do número 1 e ir para o número 3 são necessários 2 jogadas. Além disto, é possível mover mais de um dos dígitos, contanto que os dígitos sejam vizinhos. Como os cadeados possuem 4 dígitos, você pode mover, por exemplo, somente o primeiro dígito, ou pode mover os dígitos 2 e 3 ao mesmo tempos, ou até mesmo mover todos os 4 dígitos em uma só jogada, contanto que mova somente uma vez.

É importante lembrar que o cadeado possui 4 dígitos em forma de roleta e, após girar o dígito 9, pode-se ir para o dígito 8 ou de volta para o dígito 0 em uma única jogada. É possível girar para ambos os lados.

Para saber se o competido efetuou as operações de maneira ótima, Maycon pediu sua ajuda para criar um programa que, dado a combinação atual do cadeado e a combinação que se deseja chegar, forneça o número de movimentos mínimos necessários de acordo com a regra do torneio.

Entrada

A entrada é composta de vários casos de testes. Cada caso de teste é representada por dois inteiros de 4 dígitos (preenchidos com zeros à esquerda) I e F que representam, respectivamente, o estado atual do cadeado e o estado em que se deseja colocá-lo. A entrada termina com I = F = 0.

Saída

Para cada caso de teste, seu programa deve imprimir três linhas. A primeira linha contém Teste T, onde T representa caso de teste, iniciando em 1. A segunda linha deverá conter um inteiro N que indica o número de jogadas mínimas necessárias para se ir do estado I para o estado F, segundo as regras do jogo. A terceira linha deve ser mantida em branco.

Exemplo:

Entrada	Saída
1111 1111 0234 3457 0234 3889 0234 4321 0 0	Teste 1 0 Teste 2 4 Teste 3 6 Teste 4 11