6ª Maratona de Programação

Facens 2010

CADERNO DE PROBLEMAS

PROBLEMA A: BEE - MOV

Arquivo: `beemov.[c/cpp/pas/java]`

Descrição do problema

Uma colmeia abriga de 60 a 80 mil abelhas. Tem uma rainha, cerca de 400 zangões, dependendo do tamanho da colméia e milhares de operárias. Se nascerem duas ou mais rainhas ao mesmo tempo, elas lutam até que uma morra. A abelha-rainha vive até 5 anos, enquanto o resto só vive 48 dias.

Apenas as abelhas fêmeas trabalham. Os machos podem entrar em qualquer colméia, ao contrário das fêmeas. A única missão dos machos é fecundar a rainha. Como as abelhas são muito organizadas, elas utilizam uma regra para visitar cada uma das células de sua colméia para manutenção. A abelha pode iniciar a visita pela célula 1 ou pela célula 2 e move-se apenas para a célula cujo número seja superior a atual (local onde está a abelha).

Há apenas um caminho para visitar a célula 1, mas duas maneiras de chegar até a célula 2: diretamente ou através da célula 1. Para alcançar a célula 3, a abelha pode ir de 1 para 2 e depois para 3, ou de 1 para 3, ou ainda de 2 para 3, isto é, há três caminhos diferentes. A figura 1 ilustra uma abelha iniciando o percurso através da colméia.

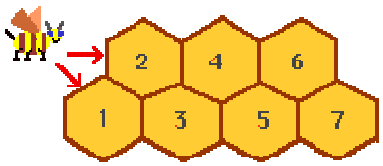


Figura 1 - Abelha iniciando sua visita

Se analisarmos com atenção, e considerarmos que se a abelha não for visitar nenhuma célula só terá 1 opção (fazer outra coisa), notaremos que o número de caminhos possíveis para visitar cada uma das células descreve a seqüência de Fibonacci, que pode ser expressa pela seguinte equação:

$$F(n) = F(n-1) + F(n-2)$$

Entrada

A entrada é constituída de vários casos de testes, inteiros positivos entre 1 e 40000 que representam o número da célula a ser visitada. A execução deve ser interrompida quando for inserida uma linha contendo apenas o valor -1.

Saída

A saída deverá conter o número de caminhos possíveis para visitar cada uma das células.

Exemplos de testes

1 2 5 10 100 -1	1 2 8 89 573147844013817084101

PROBLEMA B: TRIÂNGULO DAS BERMUDAS

Arquivo: `bermudas.[c/cpp/pas/java]`

Descrição do problema

Programação tem algo de magia negra, e computação em geral é uma ciência meio mística. Você que é conhecido como um engenheiro sobrenatural, foi convocado a auxiliar em um problema muito antigo dos navegadores: o triângulo das bermudas.

A idéia é que um navio, munido de GPS, use seu software para identificar o exato momento em que entrou na temível zona marítima. Seu programa receberá continuamente três vetores indicando os três vértices do triângulo das bermudas, e um vetor indicando o ponto onde está o navio, informando rapidamente se o navio está em perigo!

Entrada

A entrada é composta de vários casos de teste. Cada caso de teste possui 8 valores, sendo os 3 primeiros pares os três vértices do triângulo e o último par o ponto onde se encontra o navio. Todas as variáveis estão dentro da faixa ($0 \leq X \leq 1.000.000$). O valor de -1 na primeira posição de uma entrada indica o final dos casos de teste.

Saída

Para cada caso de teste será apresentada a resposta indicando se o nvaio está ou não no triângulo das bermudas.

Exemplos de testes

Entrada	**Saída**
0 0 1000 1000 1000 0 10 1 0 0 1000 1000 1000 0 1 10 0 0 1000 1000 2000 1000 1000 1000 0 0 1000 1000 2000 1000 1000 800 0 0 1000 1000 2000 1000 1000 600 0 0 1000 1000 2000 1000 1000 400 0 0 1000 1000 2000 1000 1000 200 0 0 1000 1000 2000 1000 1000 0 -1	sim nao sim sim sim nao nao nao

PROBLEMA C: THE BIG BANG THEORY

Arquivo: `bigbang.[c/cpp/pas/java]`

Descrição do Problema

No episódio 4 da primeira temporada de The Big Bang Theory Leonard compra uma réplica de uma máquina do tempo pela internet. Ao movê-la para o apartamento os garotos acidentalmente atrasam Penny para o trabalho, fazendo com que ela discuta com Leonard por suas atitudes nerd. Para que a máquina do tempo funcionasse, era preciso inserir uma seqüência numérica. Dada uma faixa de valores a seqüência é composta por números inteiros naturais com exceção dos números que compõe a série de fibonnacci (1 1 2 3 5 8 ...). Chateado com a discussão com a Penny, Leonard decide se livrar de sua coleção de coisas de nerd e os garotos tentam convencê-lo a desistir. Sheldon, preocupado com o fato de Leonard se livrar da máquina do tempo, tem alguns sonhos para decifrar o código, todos incluindo Morlocks do filme A Máquina do Tempo tentando devorá-lo.

Entrada

A entrada é constituída por casos de teste no seguinte formato: dois números M, N inteiros positivos ($0 \leq M,N \leq 100$), separados por espaço, que compõem a faixa de valores. As entradas são finalizadas quando digitado um valor menor ou igual a zero.

Saída

Para cada entrada é produzida uma saída com a seqüência numérica do código que faz com que a máquina do tempo funcione. Se nenhum número fizer parte da seqüência deverá ser impresso ERRO. Também deverá ser impresso ERRO caso o usuário não informe o início da faixa menor que o fim da faixa.

Exemplos de Testes

Entrada	**Saída**
2 1 1 3 3 7 1 10 20 30 -1 -1	ERRO ERRO 4 6 7 4 6 7 9 10 20 22 23 24 25 26 27 28 29 30

PROBLEMA D: THEY CALL ME THE CLEANER

Arquivo: `cleaner.[c/cpp/pas/java]`

Descrição do problema

Engenheiros são certamente uma raça mal compreendida. Freqüentemente são menosprezados em relação aos seus colegas vendedores, mais importantes aos olhos das grandes corporações. Em outras ocasiões, são vistos como solucionadores universais de problemas aleatórios: “Mas você é engenheiro, não pode fazer nada com esse buraco na camada de ozônio?”.

Um dos seus colegas de trabalho, conhecido como “the cleaner”, num destes momentos místicos, lhe lançou um desafio. Ele gostaria de realizar o seu trabalho de limpeza de forma melhor. Para tal, ele não poderia passar com seu esfregão duas vezes no mesmo corredor. Então ele lhe solicitou que usasse seus conhecimentos em engenharia para lhe dizer se era possível, dado um mapa, passar por todos os corredores executando a limpeza sem repetir em nenhum momento um corredor.

Cansado de ouvir pedidos de soluções descabidas em relação a sua formação, e vendo um problema que faz sentido para você, você decidiu ajudar. Mais ainda, disse ao colega que ele pode informar qualquer mapa para você, e você será capaz de dizer se é possível ou não realizar a limpeza otimizada. Os mapas serão informados através das intersecções entre os corredores.

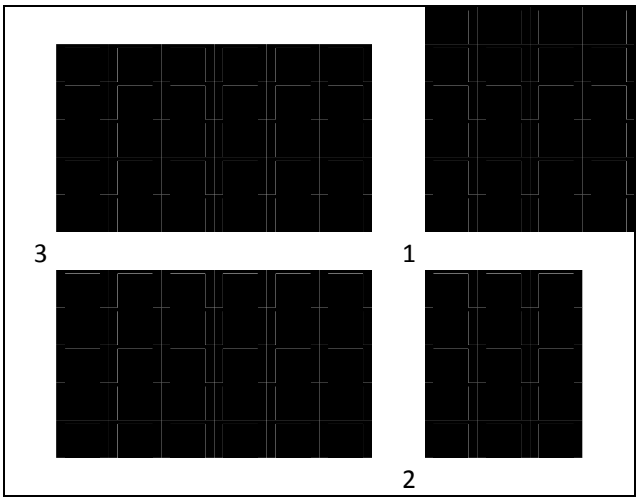


Figura 1: Exemplo de mapa

No mapa da figura 1 pode-se ver 3 intersecções em corredores. São consideradas somente intersecções onde existe a opção de mudar de direção. O canto, por exemplo, só existe um caminho e, portanto, não é considerada uma intersecção.

Entrada

A entrada é composta de vários casos de teste. A primeira linha dos casos de teste indica o número de corredores M e o número de intersecções entre os corredores N ($3 \leq M,N \leq 100$). Em seguida são informadas N linhas, cada uma indicando X, Y e Z sendo que X é a intersecção de onde parte um corredor, Y é a intersecção onde chega um corredor e Z é o número de corredores que ligam estas duas intersecções ($3 \leq X,Y \leq N$) ($1 \leq Z \leq 10$).

Saída

Para cada caso de teste será apresentada a resposta indicando se existe ou não um caminho capaz de percorrer todos os corredores sem repetições.

Exemplos de testes

Entrada	**Saída**
3 3 1 2 2 1 3 2 2 3 1 5 7 1 2 2 1 3 2 2 3 1 1 4 1 2 4 1 1 5 1 3 5 1 0	sim nao

PROBLEMA E: COLUNAS DE CONCRETO

Arquivo: `concreto.[c/cpp/pas/java]`

Descrição do problema

Uma empresa de construção civil precisa construir um software para calcular o volume de concreto necessário encher suas vigas de sustentação. As informações necessárias são: quantidade de vigas por linha e por coluna, largura da viga, altura da viga e o raio dos cantos da mesma, caso exista um acabamento com cantos arredondados. As 4 últimas medidas são dadas em metros (m).

A partir dessas informações o programa deve retornar o volume necessário (m^3) de concreto que deve ser usado para preencher as vigas.

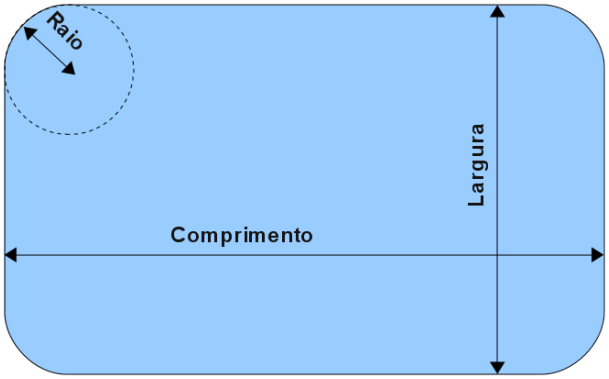


Figura 1: Corte da viga

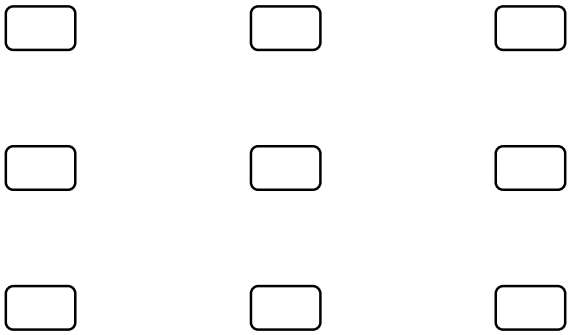


Figura 2: Visão de cima das vigas

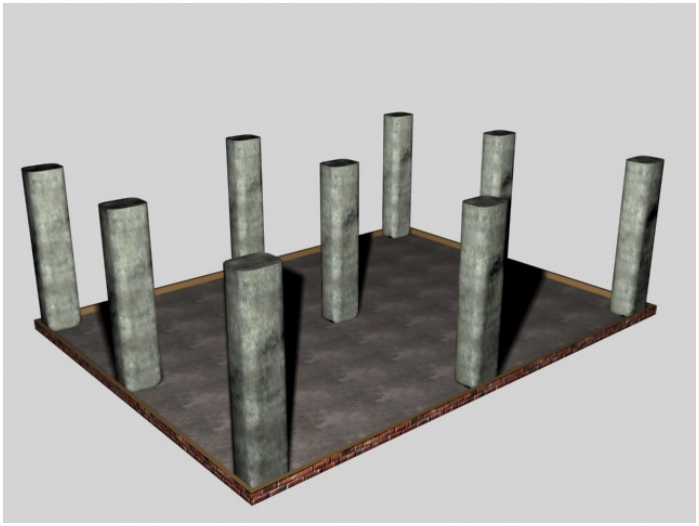


Figura 3: Visão 3D do espaço

Entrada

O programa deve receber, nessa ordem, o número de linhas (entre 1 e 9, 3 no exemplo), o número de colunas (entre 1 e 9, 2 no exemplo), a largura total da coluna (em metros, entre 0.1 e 9.9), o comprimento total da viga (em metros, entre 0.1 e 9.9), a altura total da viga (em metros, entre 0.5 e 9.9) e o raio dos cantos da viga (em metros, entre 0 e 4.9). Se necessário, deve-se considerar o número PI como 3,1416.

O programa deve receber, nessa ordem, o número de linhas (3 no exemplo), o número de colunas (2 no exemplo), a largura total da coluna (em metros), o comprimento total da viga(em metros), a altura total da viga (em metros) e o raio do canto da viga (em metros). Os casos de teste terminam se o número de linhas for zero.

Saída

O programa deve retornar o volume de concreto (em metros cúbicos) com uma casa decimal de precisão.

Exemplos de testes

Entrada	**Saída**
1 1 2 3 5 0.1	30.0
2 3 2 4 9 0.5	420.4

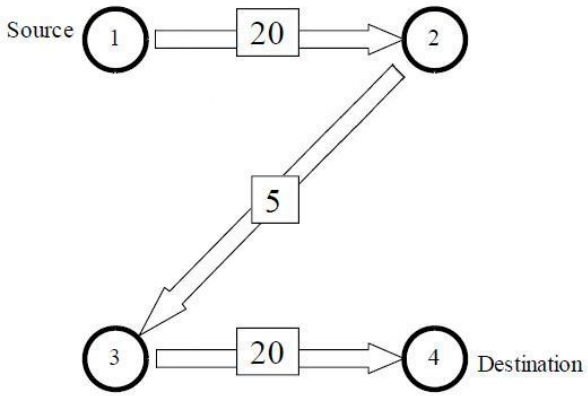
PROBLEMA F: ROTEAMENTO NA INTERNET

Arquivo: `internet.[c/cpp/pas/java]`

Descrição do Problema

Na Internet, computadores e outros dispositivos de rede estão amplamente conectados, portanto, muitos caminhos podem existir entre um par de nós da rede. Muitas vezes, existe a necessidade de saber qual é o custo (tempo) para transmitir um pacote de dados de um nó a outro da rede através de um caminho específico. Ao transmitir um pacote de dados via Internet, certamente ele passará por muitos roteadores (nós) até que chegue ao seu destino final, então, para calcular o tempo total necessário para transmissão de um pacote nesse cenário, soma-se os tempos parciais gastos para transmitir o pacote entre nós intermediários da rede.

Por exemplo, a figura abaixo mostra uma rede com quatro nós (círculos) e com três conexões entre eles. Cada conexão está rotulada com o tempo necessário para que um pacote seja transmitido entre os nós ligados pela conexão.



No exemplo, o tempo total para transmitir um pacote do nó 1 para o nó 4 é igual a 45ms.

Uma empresa multinacional, com filiais espalhadas pelo mundo, está com problemas de performance na comunicação entre suas plantas. Tal empresa contratou sua equipe para desenvolver um programa que calcule o tempo total necessário para transmissão de um pacote entre dois nós de sua rede.

Entrada

O arquivo de entrada contém descrições de várias redes. Cada descrição começa com uma linha contendo os rótulos de todos os nós da rede separados por espaço. Esses rótulos podem ser números, letras ou palavras e servem como identificadores dos

nós. As próximas linhas contêm descrições das conexões entre os nós da rede. Cada uma dessas linhas possui o rótulo do nó de origem, o rótulo do nó de destino e o tempo de transmissão de um pacote entre os nós. O tempo de transmissão é um número inteiro maior ou igual a 1 e menor ou igual a 20. Caso dois nós sejam duplamente conectados, as conexões serão descritas em duas linhas distintas. Uma linha contendo o número zero termina as descrições das conexões. A última linha da descrição de uma rede contém o caminho que o pacote de dados deve percorrer, ou seja, apresenta uma seqüência de nós da origem ao destino. O arquivo de entrada é finalizado por uma linha que contém a sigla EOI (*End Of Input*).

Saída

Para cada descrição de rede, o programa desenvolvido deverá apresentar o tempo total para que um pacote de dados termine o percurso descrito ou deverá imprimir "percurso incorreto" para percursos que não possam ser completados.

Exemplo de Entrada	**Exemplo de Saída**
1 2 3 4 1 2 20 2 3 5 3 4 20 0 1 2 3 4 a b c d a c 10 c b 5 b d 10 0 a c b d 1 2 3 4 1 2 20 2 3 5 3 4 20 0 1 3 2 4 EOI	45 25 percurso incorreto

PROBLEMA G: iPASS

Arquivo: `ipass.[c/cpp/pas/java]`

Descrição do Problema

Um milionário indiano da área de tecnologia, preocupado com a segurança de seus bens pessoais (jóias, relógios, moedas etc) resolveu criar um dispositivos eletrônico para gerar a sequência numérica para abertura de seus cofres.

O dispositivo funciona basicamente através de rotação de linhas e colunas de uma matriz de tamanho pré-determinado, sendo a sequência numérica das linhas a senha para abrir um cofre.

A imagem 1 mostra a matriz original, a imagem 2 exemplifica um descolamento a direita da linha 1, a imagem 3 o deslocamento de uma posição da linha 2 para esquerda, a imagem 4 o deslocamento de duas posições da coluna 2 para cima e a imagem 5 o descolamento da de uma posição da coluna 3 para baixo.

1	2	3
4	5	6
7	8	9

Figura 2 - Matriz Original

3	1	2
4	5	6
7	8	9

Figura 3 - Matriz com descolamento a direita

1	2	3
5	6	4
7	8	9

Figura 4 - Matriz com deslocamento a esquerda

1	8	3
4	2	6
7	5	9

Figura 5 - Matriz com deslocamento da coluna 2 para cima

1	2	9
4	5	3
7	8	6

Figura 6 - Matriz com deslocamento da coluna 3 para baixo

Entrada

A entrada é composta de vários casos de testes, sendo os dois primeiros números L e C que representam as dimensões da matriz (0 < linha, coluna < 10), seguido do número de deslocamentos da matriz D (0 <= D <= 500) e de uma linha com: l ou c (linha ou coluna), número da linha ou coluna, número do deslocamento (0 <= R <= 100) seguido do sentido do deslocamento (d - direita ou e - esquerda para linhas / c - cima ou b - baixo para colunas). Os casos de testes terminam quando as dimensões da matriz

forem -1 -1. O preenchimento da matriz é sempre seqüencial, iniciando pelo valor 1 na posição 1,1, 2 na posição 1,2, 3 na posição 1,3, assim por diante.

Saída

Para cada caso de testes, a saída deverá conter os números da matriz separados por espaço, seguido de uma quebra de linha.

Exemplos de Testes

3 3 2 l 1 5 d c 1 0 e 2 2 4 l 1 1 e l 1 1 d c 1 2 c l 2 2 d -1 -1	2 3 1 4 5 6 7 8 9 1 2 3 4

PROBLEMA H: MM

Arquivo: `mm.[c/cpp/pas/java]`

Descrição do problema

Um famoso engenheiro de computação que venceu várias maratonas chamou sua equipe para ajudá-lo em um problema. Aparentemente, isso é um teste para que ele o contrate na sua equipe de desenvolvimento.

Ele quer gerar códigos aleatórios em um sistema de acordo com uma determinada máscara. Além de se preocupar com uma máscara, ele ainda deseja poder configurar uma classe de caracteres que fará parte desse sistema.

Entrada

O sistema a ser desenvolvido deve receber um inteiro 0 <= M <= 10 indicando o número de cadeia de caracteres a ser informado. Para cada uma das M linhas deve-se receber um inteiro 0 <= N <= 10 indicando o número de caracteres que compõe essa cadeia e uma entrada de texto em seguida contendo N caracteres. Essa cadeia representa uma base de numeração, e cada letra da cadeia um algarismo desta base de numeração.

Após a entrada das bases de numeração deve-se receber um texto indicando uma máscara a ser usada. Essa máscara é composta por dígitos que representam o índice da cadeia de caracteres da base de numeração a ser usado na posição.

Dadas essas configurações o sistema espera por um inteiro 0 <= K <= 1000 indicando o número de casos de teste a realizar, seguido por K inteiros, representando cada um dos casos de teste.

Saída

A saída a ser exibida é o valor do número convertido para a base numérica usando a máscara informada. Caso a máscara não suporte tal número, exibir a mensagem "Nao suportado pela mascara".

Exemplos de testes

Entrada	**Saída**
2 2 01	0aa0 0ac1 Nao suportado pela mascara

3 abc 0110 4 0 5 37 18 1 4 a1b2 00000 5 0 1 2 3 4 0	1aa0 11111 11112 1111b 11112 11121

PROBLEMA I: PLACAR

Arquivo: `placar.[c/cpp/pas/java]`

Descrição do problema

Futebol é um esporte que tem lugar especial no coração da maioria dos brasileiros. Devido à popularidade desse esporte, são feitos investimentos e pesquisas para melhorar a sistemática das partidas, o desempenho dos jogadores e a atuação dos árbitros. Dentro desse contexto, uma confederação de futebol contratou a sua equipe para desenvolver um programa que a partir do placar final de um jogo de futebol, determine todas as possíveis sucessões de gols marcados na partida. Por exemplo, para uma partida entre duas equipes A x B com placar final de 3 x 1, as possíveis sucessões de gols são "AAAB", "AABA", "ABAA" e "BAAA".

Entrada

A entrada é composta de vários casos de teste. Cada caso de teste possui quatro valores, sendo os dois primeiros os nomes das equipes e os dois últimos os gols marcados por elas respectivamente. Cada nome de equipe é representado por uma letra maiúscula (A-Z). O número de gols marcados por uma equipe pertence ao intervalo [0, 7]. O valor -1 na primeira posição de uma entrada indica o final dos casos de teste.

Saída

Para cada caso de teste deverão ser apresentadas todas as possíveis sucessões de gols marcados, uma por linha e em ordem alfabética.

Exemplos de testes

Entrada	**Saída**
A B 0 0 C D 3 1 F E 2 2 -1	Nao houveram gols CCCD CCDC CDCC DCCC EEFF EFEF EFFE FEEF FEFE FFEE

PROBLEMA J: PROCESSAMENTO DE TEXTO

Arquivo: `proctexto.[c/cpp/pas/java]`

Uma empresa de hardware está concluindo o seu driver de teclado e para isso está requisitando que a sua equipe construa um simulador dos sinais enviados pelo teclado, exibindo a saída do texto processado no bloco de notas.

Para este protótipo a empresa reduziu o escopo emitindo apenas alguns dos sinais de teclado especiais que devem ser tratados, conforme descrito abaixo:

Sinal	**Detalhe**
[SHIFT]	Aplica alteração de maiúsculo/minúsculo no próximo caracter escrito
[CAPS]	Liga/Desliga o caps lock, indicando que as letras digitadas agora são maiúsculas
[BKSPC]	Apaga o último caracter digitado

Como nem todos os padrões foram definidos deve-se assumir que não há controle de caracteres maiúsculos/minúsculos no envio do sinal, podendo assim ser recebido de qualquer maneira. Sinais inválidos serão processados como texto e exibidos na saída do programa. Todo o processamento de maiúsculos e minúsculos é feito apenas pelos sinais [SHIFT] e [CAPS].

Entrada

A entrada do programa é composta de um inteiro N ($0 < N \leq 1000$), que representa o número de cenários de teste a serem executados. Cada cenário de teste é composto por uma string S com até 5000 caracteres minúsculos, contendo os dados recebidos pelo teclado.

Saída

A saída é determinada pela exibição do texto após o processamento dos sinais.

Exemplos de testes

Entrada	Saída
2 [SHIFT]ola,[SHIFT]bem vindo ao[CAPS] console de testes[CAPS]! [teste][Shift]bom dia![CAPS]corrigirr[BKSPC][CAPS]	Ola, Bem vindo ao CONSOLE DE TESTES! [teste]Bom dia!CORRIGIR

PROBLEMA K: SUDOKU SOLVER

Arquivo: `sudoku.[c/cpp/pas/java]`

Descrição do problema

Sudoku é um famoso quebra-cabeça baseado na colocação de números. O objetivo do jogo é a colocação de números de 1 a 9 em cada uma das células vazias numa grade de 9×9, constituída por 3×3 subgrades chamadas regiões. O quebra-cabeça contém algumas pistas iniciais. Cada coluna, linha e região só pode ter um número de cada um dos de 1 a 9. O quebra-cabeça está completo quando todas as posições estão preenchidas com números, baseados nas regras.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figura 1: Quebra-cabeça no estado inicial

A sua missão é construir um programa para resolver esse tipo de quebra-cabeça de nível fácil. Nessa modalidade, é possível encontrar todos os números em um processo iterativo que consiste em 3 verificações: Verificar o único valor possível em uma determinada célula, verificar a única posição possível de um número em uma linha ou coluna, verificar a única posição possível de um número em uma das 9 regiões 3x3. No nível fácil, nenhuma verificação adicional é necessária a cada passo, para preencher todo o quebra-cabeça. Sempre haverá uma célula que poderá ser preenchida com base em 1 das 3 verificações.

Entrada

9 linhas, contendo 9 números cada. Os números entre 1 e 9 correspondente às dicas nas posições das células corretas e 0 que correspondem às células em branco. Para encerrar, apenas um 'X' na primeira linha.

Saída

9 linhas, contendo 9 números cada, correspondendo a solução final do quebra-cabeça.

Exemplos de teste

Entrada	**Saída**
3 0 8 0 4 0 0 6 0 9 0 0 0 3 0 0 7 8 0 6 0 0 0 2 0 3 0 0 5 0 0 0 0 7 2 0 0 3 0 5 0 9 0 1 0 0 8 4 0 0 0 0 5 0 0 2 0 4 0 0 0 9 0 8 9 0 0 5 0 0 0 1 0 4 0 0 9 0 3 0 2 3 0 8 0 4 0 0 6 0 9 0 0 0 3 0 0 7 8 0 6 0 0 0 2 0 3 0 0 5 0 0 0 0 7 2 0 0 3 0 5 0 9 0 1 0 0 8 4 0 0 0 0 5 0 0 2 0 4 0 0 0 9 0 8 9 0 0 5 0 0 0 1 0 4 0 0 9 0 3 0 2 x	3 7 8 9 4 1 2 6 5 9 1 2 6 3 5 4 7 8 4 6 5 8 7 2 1 3 9 1 5 9 3 8 4 7 2 6 7 3 6 5 2 9 8 1 4 2 8 4 1 6 7 9 5 3 6 2 3 4 1 8 5 9 7 8 9 7 2 5 3 6 4 1 5 4 1 7 9 6 3 8 2 3 7 8 9 4 1 2 6 5 9 1 2 6 3 5 4 7 8 4 6 5 8 7 2 1 3 9 1 5 9 3 8 4 7 2 6 7 3 6 5 2 9 8 1 4 2 8 4 1 6 7 9 5 3 6 2 3 4 1 8 5 9 7 8 9 7 2 5 3 6 4 1 5 4 1 7 9 6 3 8 2

PROBLEMA L: O UNIVERSO NUMA CASCA DE NOZ

Arquivo: `universo.[c/cpp/pas/java]`

Descrição do problema

Um grupo de engenheiros e físicos está criando uma sala especial para testar conceitos avançados de física. Nesta sala, teremos dois tipos de dispositivos revolucionários: os terminais de teleporte e os campos de gravidade aumentada.

Os terminais de teleporte, como o próprio nome já diz, transportarão um corpo instantaneamente a outro determinado ponto da sala. Uma vez que o corpo chegue ao terminal de teleporte ele é imediatamente levado ao ponto de destino daquele terminal. Os campos de gravidade aumentada são regiões que simulam uma força gravitacional aumentada. Entrar nestas posições é simples, mas sair envolve um esforço maior, proporcional ao aumento do campo gravitacional.

Tendo em vista que a tecnologia é altamente instável, surgiu a proposta de criar um veículo auto guiado para trafegar pela sala, passando pelos terminais e pelos campos, para observar os efeitos colaterais. Este veículo possuirá as informações da sala, dos obstáculos, dos terminais e dos campos. Finalmente, um veículo que se move de forma aleatória não permite testes muito conclusivos, então se decidiu que ele terá que encontrar um objeto na sala, realizando o caminho mais rápido possível. Sua equipe foi convocada para modelar o software deste veículo.

Entrada

A entrada é composta de vários casos de teste. A primeira linha dos casos de teste indicam as dimensões MxN da sala ($3 \leq M,N \leq 10$), e as coordenadas X,Y da posição do veículo auto guiado e em seguida a posição Z,W do objeto que ele deve coletar ($1 \leq X \leq M$, $1 \leq Y \leq N$, $1 \leq Z \leq M$, $1 \leq W \leq N$, $X \neq Z$ ou $Y \neq W$). Um caso de teste em que M é zero indica o final dos casos de teste.

Em seguida, é apresentada uma matriz MxN com os setores da sala. O conteúdo de cada setor é identificado com um valor numérico T ($-100 \leq T \leq 10$). O valor zero indica que a posição é um obstáculo (parede), onde o veículo não pode se mover. Valores positivos indicam o tempo que leva para o veículo sair do setor, tendo em vista o campo gravitacional. Valores negativos indicam a existência de um terminal de teleporte. O número negativo indica o setor onde o veículo irá aparecer caso entre no terminal de

teleporte. Os setores são numerados a partir de 1, da esquerda para a direita e de cima para baixo.

Saída

Para cada caso de teste será apresentado o tempo total que o veículo levaria para realizar o caminho mais rápido, seguido dos setores por onde ele passaria para fazer o caminho mais rápido.

Exemplos de testes

Entrada	**Saída**
3 4 1 1 3 4 1 3 -8 1 1 0 0 1 2 2 1 1 5 5 3 3 5 5 1 1 1 1 1 0 0 0 0 1 -4 1 1 0 1 0 0 0 0 0 1 1 1 1 1 10 10 1 1 10 10 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1 -77 1 0 0 1 1 0 1 1 1 1 1 0 0 2 1 0 1 1 1 1 1 0 0 2 1 0 1 1 1 1 1 0 0 2 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 -90 0 0 1 1 0 0 0 0 0 0 0 1 1 4 4 1 1 1 1 1 1 1 1 0	5 1, 2, 3, 8, 12 caminho inexistente 8 1, 2, 3, 4, 5, 6, 16, 26, 77, 90, 100