8^o Treino para Alunos da UFPR

15 de Junho de 2013

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

Organizadores:
Vinicius Kwiecien Ruoso e Ricardo Tavares de Oliveira

Lembretes:

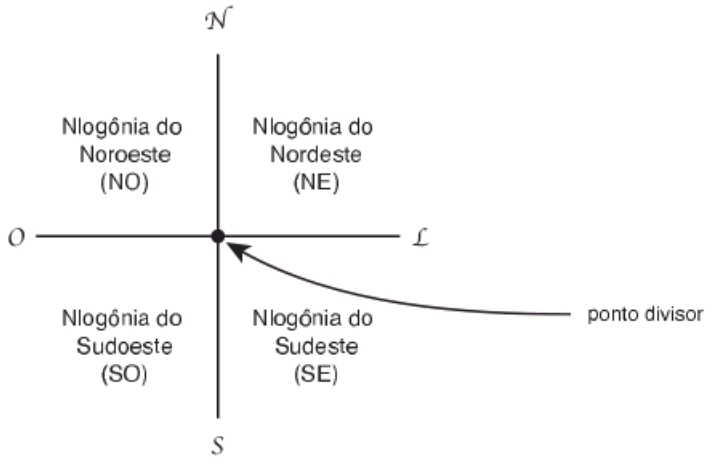
- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema 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: Divisão da Nlogônia

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

Depois de séculos de escaramuças entre os quatro povos habitantes da Nlogônia, e de dezenas de anos de negociações envolvendo diplomatas, políticos e as forças armadas de todas as partes interessadas, com a intermediação da ONU, OTAN, G7 e SBC, foi finalmente decidida e aceita por todos a maneira de dividir o país em quatro territórios independentes.

Ficou decidido que um ponto, denominado *ponto divisor*, cujas coordenadas foram estabelecidas nas negociações, definiria a divisão do país da seguinte maneira. Duas linhas, ambas contendo o ponto divisor, uma na direção norte-sul e uma na direção leste-oeste, seriam traçadas no mapa, dividindo o país em quatro novos países. Iniciando no quadrante mais ao norte e mais ao oeste, em sentido horário, os novos países seriam chamados de Nlogônia do Noroeste, Nlogônia do Nordeste, Nlogônia do Sudeste e Nlogônia do Sudoeste.



A ONU determinou que fosse disponibilizada uma página na Internet para que os habitantes pudessem consultar em qual dos novos países suas residências estão, e você foi contratado para ajudar a implementar o sistema.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro K indicando o número de consultas que serão realizadas ($0 < K \leq 10^3$). A segunda linha de um caso de teste contém dois números inteiros N e M representando as coordenadas do ponto divisor ($-10^4 < N, M < 10^4$). Cada uma das K linhas seguintes contém dois inteiros X e Y representando as coordenadas de uma residência ($-10^4 \leq X, Y \leq 10^4$). Em todas as coordenadas dadas, o primeiro valor corresponde à direção leste-oeste, e o segundo valor corresponde à direção norte-sul.

O final da entrada é indicado por uma linha que contém apenas o número zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma linha contendo:

- a palavra *divisa* se a residência encontra-se em cima de uma das linhas divisórias (norte-sul ou leste-oeste);
- *NO* se a residência encontra-se na Nlogônia do Noroeste;
- *NE* se a residência encontra-se na Nlogônia do Nordeste;
- *SE* se a residência encontra-se na Nlogônia do Sudeste;
- *SO* se a residência encontra-se na Nlogônia do Sudoeste.

Exemplo de entrada	**Exemplo de saída**
3 2 1 10 10 -10 1 0 33 4 -1000 -1000 -1000 -1000 0 0 -2000 -10000 -999 -1001 0	NE divisa NO divisa NE SO SE

B: Zak Galou

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

Zak Galou é um famoso bruxo matador de monstros. Diz a lenda que existe uma caverna escondida nos confins da selva contendo um tesouro milenar. Até hoje nenhum aventureiro conseguiu recuperar o tesouro, pois ele é bem guardado por terríveis monstros. Mas Zak Galou não é um aventureiro qualquer e decidiu preparar-se para recuperar o tão sonhado tesouro.

Zak Galou dispõe de uma certa quantidade de mana (uma espécie de energia mágica) e de uma lista de M magias. Cada monstro tem um determinado número de pontos de vida. Cada vez que Zak Galou lança uma magia contra um monstro, Zak gasta uma certa quantidade de mana (o custo da magia) e inflinge um certo dano ao monstro. O dano inflingido provoca a perda de pontos de vida do monstro (o número de pontos perdidos depende da magia). Um monstro está morto se tiver zero ou menos pontos de vida. Zak sempre luta contra um monstro a cada vez. Como é um bruxo poderoso, ele pode usar a mesma magia várias vezes, desde que possua a quantidade necessária de mana.

Em suas pesquisas, Zak Galou conseguiu o mapa do tesouro. A caverna é representada como um conjunto de salões conectados por galerias. Os salões são identificados sequencialmente de 1 a N. Zak sempre inicia no salão 1 e o tesouro está sempre no salão N. Existem K monstros identificados seqüencialmente de 1 a K. Cada monstro vive em um salão, do qual não sai (note que é possível que mais de um monstro viva no mesmo salão). Durante a busca pelo tesouro, Zak Galou pode sair ou recuperar o tesouro de um salão somente se o salão estiver vazio (sem monstro). Em outras palavras, Zak deve sempre, antes de sair ou de recuperar o tesouro de um salão, matar o(s) monstro(s) que lá viver(em).

Dadas as descrições das magias, dos monstros e da caverna, sua tarefa é determinar a quantidade mínima inicial de mana necessária para que Zak Galou consiga recuperar o tesouro.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém quatro inteiros M, N, G e K, indicando respectivamente o número de magias ($1 \leq M \leq 1000$), de salões ($1 \leq N \leq 1000$), de galerias ($0 \leq G \leq 10^6$) e de monstros ($0 \leq K \leq 1000$).

Cada uma das M linhas seguintes descreve uma magia. A descrição de uma magia contém dois números inteiros, a quantidade de mana consumida (entre 1 e 1000) e o número de pontos de danos provocados (também entre 1 e 1000).

Em seguida, há G linhas, cada uma descrevendo uma galeria. Uma galeria é descrita por dois números inteiros A e B ($A \neq B$), representando os salões que a galeria conecta. Zak pode utilizar a galeria nos dois sentidos, ou seja, para ir de A para B ou de B para A.

Finalmente, as últimas K linhas de um caso de teste descrevem os monstros. A descrição de um monstro contém dois números inteiros representando respectivamente o salão no qual ele vive (entre 1 e N inclusive) e o seu número inicial de pontos de vida (entre 1 e 1000 inclusive).

O final da entrada é indicado por $M = N = G = K = 0$.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída contendo um número inteiro, a quantidade mínima inicial de mana necessária. Caso não seja possível recuperar o tesouro, você deve imprimir -1.

Exemplo de Entrada	**Exemplo de Saída**
3 4 4 2 7 10 13 20 25 50 1 2 2 4 1 3 3 4 2 125 3 160 3 4 4 1 7 10 13 20 25 50 1 2 2 4 1 3 3 4 2 125 1 3 1 1 1000 1000 1 2 3 1000 0 0 0 0	70 0 -1

C: Alarme Despertador

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

Daniela é enfermeira em um grande hospital, e tem os horários de trabalho muito variáveis. Para piorar, ela tem sono pesado, e uma grande dificuldade para acordar com relógios despertadores.

Recentemente ela ganhou de presente um relógio digital, com alarme com vários tons, e tem esperança que isso resolva o seu problema. No entanto, ela anda muito cansada e quer aproveitar cada momento de descanso. Por isso, carrega seu relógio digital despertador para todos os lugares, e sempre que tem um tempo de descanso procura dormir, programando o alarme despertador para a hora em que tem que acordar. No entanto, com tanta ansiedade para dormir, acaba tendo dificuldades para adormecer e aproveitar o descanso.

Um problema que a tem atormentado na hora de dormir é saber quantos minutos ela teria de sono se adormecesse imediatamente e acordasse somente quando o despertador tocasse. Mas ela realmente não é muito boa com números, e pediu sua ajuda para escrever um programa que, dada a hora corrente e a hora do alarme, determine o número de minutos que ela poderia dormir.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é descrito em uma linha, contendo quatro números inteiros $H1$, $M1$, $H2$ e $M2$, com $H1 : M1$ representando a hora e minuto atuais, e $H2 : M2$ representando a hora e minuto para os quais o alarme despertador foi programado ($0 \leq H1 \leq 23, 0 \leq M1 \leq 59, 0 \leq H2 \leq 23, 0 \leq M2 \leq 59$).

O final da entrada é indicado por uma linha que contém apenas quatro zeros, separados por espaços em branco.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma linha, cada uma contendo um número inteiro, indicando o número de minutos que Daniela tem para dormir.

Exemplo de entrada	**Exemplo de saída**

`1 5 3 5`
`23 59 0 34`
`21 33 21 10`
`0 0 0 0`
	`120`
	`35`
	`1417`

D: 4 values whose sum is 0

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

The SUM problem can be formulated as follows: given four lists A, B, C, D of integer values, compute how many quadruplet $(a, b, c, d) \in A \times B \times C \times D$ are such that $a + b + c + d = 0$. In the following, we assume that all lists have the same size n.

Input

The input contains several test cases. For each test case the first line of the input contains the size of the lists N (this value can be as large as 4000). We then have N lines containing four integer values (with absolute value as large as 2^{28}) that belong respectively to A, B, C and D.

Output

For each test case your program has to write on line with the number of quadruplets whose sum is zero. The last test case is followed by a line containing one zero.

Exemplo de entrada	**Exemplo de saída**
6 -45 22 42 -16 -41 -27 56 30 -36 53 -37 77 -36 30 -75 -46 26 -38 -10 62 -32 -54 -6 45 0	5

E: Irmãos

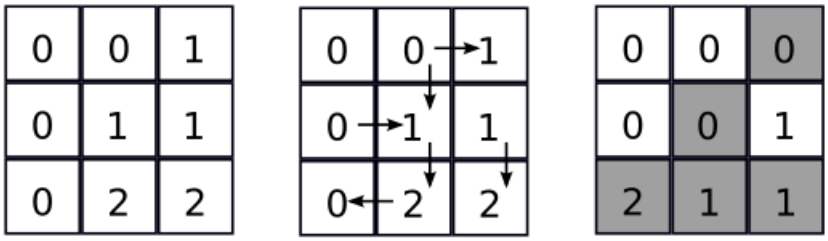
File: irmaos.[c|cpp|java|pas]

O reino de ACM era comandado por um grande Rei que era obcecado por ordem. O reino tinha forma retangular, e o rei dividiu o território em uma grade de pequenos países retangulares. Antes de morrer, o Rei distribuiu os países entre seus filhos.

No entanto, ele não sabe que seus filhos desenvolveram uma estranha rivalidade; o primeiro herdeiro odiava o segundo herdeiro, mas não os demais; o segundo herdeiro odiava o terceiro herdeiro, mas não os demais, e assim por diante. Finalmente, o último herdeiro odiava o primeiro herdeiro, mas não os demais.

Logo que o Rei morreu, essa estranha rivalidade entre os filhos do Rei desencadeou uma guerra generalizada no reino. Os ataques ocorriam apenas entre pares de países adjacentes (que compartilham uma borda vertical ou horizontal). Um país X atacava um país adjacente Y sempre que o proprietário de X odiava o proprietário de Y. O país atacado era sempre conquistado pelo irmão atacante. Por uma questão de honra, todos os ataques aconteciam simultaneamente, e um conjunto de ataques simultâneos recebia o nome de batalha. Após um certo número de batalhas, os filhos sobreviventes assinaram um tratado de trégua e nunca mais batalharam.

Por exemplo, se o Rei tinha três filhos (0, 1 e 2) a figura abaixo mostra o que acontece na primeira batalha dada uma distribuição inicial das terras:



Terras antes da batalha. | Ataques durante a batalha. | Terras após a batalha. Alterações em cinza.

Você foi contratado para ajudar um historiador da ACM a determinar, dados o número de herdeiros, a distribuição inicial de territórios e o número de batalhas, qual a distribuição final de territórios após as batalhas.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros N, R, C e K separados por um espaço em branco. N é o número de herdeiros ($2 \leq N \leq 100$), R e C são as dimensões do reino ($2 \leq R, C \leq 100$) e K é o número de batalhas ($1 \leq K \leq 100$). Herdeiros são identificados por inteiros de 0 até N-1. Cada uma das seguintes R linhas contém C inteiros $H_{r,c}$ separados por um espaço em branco, representando a distribuição inicial de terras: $H_{r,c}$ é o proprietário inicial do país na linha r e coluna c ($0 \leq H_{r,c} \leq N - 1$).

O último caso de teste é seguido de uma linha contendo quatro zeros separados por um espaço em branco.

Saída

Para cada caso de teste, seu programa deve imprimir R linhas com C inteiros cada, separados por um espaço em branco, no mesmo formato da entrada, representando a distrubuição das terras após todas as batalhas.

Exemplo de entrada	**Exemplo de saída**
3 4 4 3 0 1 2 0 1 0 2 0 0 1 2 0 0 1 2 2 4 2 3 4 1 0 3 2 1 2 8 4 2 1 0 7 1 6 2 5 3 4 0 0 0 0	2 2 2 0 2 1 0 1 2 2 2 0 0 2 0 0 1 0 3 2 1 2 7 6 0 5 1 4 2 3