CI067 – Oficina de Computação
Exercícios # 08
1º semestre 2010

Em todos os exercícios abaixo, os programas possuem um argumento de linha de comando opcional. Se este argumento não é fornecido, a entrada de dados é lida da Entrada Padrão, e a saída de dados é feita para a Saída Padrão. Caso o argumento seja fornecido na linha de comando, este representa o nome do arquivo de onde devem ser lidos os dados de entrada. A saída de dados deve ser feita para um arquivo cujo nome é composto pelo nome do arquivo de entrada acrescido do sufixo `-out`.

1. **(pizza)** Rodrigo pediu uma pizza de mussarela de N fatias, uma parte somente com cebola e o resto somente com azeitonas. Entretanto, ao receber a pizza em casa, notou que o motoqueiro que a entregou não foi cuidadoso o suficiente, pois tanto as tiras de cebola quanto as azeitonas estavam espalhadas por toda a pizza. Para piorar, como a pizza era de mussarela, as tiras de cebola e as azeitonas estavam grudadas na pizza.
   Como gosta mais de cebola do que de azeitona, Rodrigo deseja pegar fatias consecutivas da pizza de tal forma que estas contenham a maior diferença possível entre tiras de cebola e azeitonas. Para isso, ele contou quantas tiras e quantas azeitonas tinham em cada fatia e subtraiu os dois valores, nessa ordem. Assim, sempre que uma fatia contiver mais cebolas que azeitonas, ela recebe um número positivo, e caso contrário, um número negativo. Uma fatia cujo número seja zero contém o mesmo número de tiras de cebolas e azeitonas.
   Por exemplo, supondo que as fatias contenham as seguintes diferenças: 5,-3,-3, 2,-1, 3, pode-se pegar uma fatia consecutiva com 9 cebolas a mais que azeitonas, utilizando as fatias com as diferenças 2,-1, 3, 5 (lembre-se de que estamos tratando de um círculo e, portanto, a fatia com diferença 5 é vizinha da fatia com diferença 3 e vice-versa).
   Repare que é melhor não escolher nenhuma fatia caso somente seja possível escolher fatias consecutivas com mais azeitonas que cebolas.

   Escreva um programa que, fornecidas as diferenças entre as quantidades de cebolas e azeitonas em cada fatia de pizza, retorne a maior quantidade possível de cebolas que Rodrigo pode comer a mais do que a quantidade de azeitonas utilizando somente fatias consecutivas de pizza. (lembrando que a primeira fatia é adjacente à última e vice-versa).

   A entrada contém um único conjunto de dados. A primeira linha da entrada contém um inteiro N que indica o número de fatias de pizza ($1 \leq N \leq 100.000$). A segunda linha contém N inteiros K ($-100 \leq K \leq 100$) separados por um espaço em branco com as diferenças entre as quantidades de cebolas e de azeitonas.

   Seu programa deve produzir uma única linha, contendo a maior quantidade de cebolas que Rodrigo pode comer a mais do que azeitonas.

2. **(marcha)** Este ano o sargento está tendo mais trabalho do que de costume para treinar os recrutas. Um deles é muito atrapalhado, e de vez em quando faz tudo errado – por exemplo, ao invés de virar à direita quando comandado, vira à esquerda, causando grande confusão no batalhão.
O sargento tem fama de durão e não vai deixar o recruta em paz enquanto este não aprender a executar corretamente os comandos. No sábado à tarde, enquanto todos os outros recrutas estão de folga, ele obrigou o recruta a fazer um treinamento extra. Com o recruta marchando parado no mesmo lugar, o sargento emitiu uma série de comandos "esquerda volver!" e "direita volver!". A cada comando, o recruta deve girar sobre o mesmo ponto e dar um quarto de volta na direção correspondente ao comando. Por exemplo, se o recruta está inicialmente com o rosto voltado para a direção norte, após um comando de "esquerda volver!" ele deve ficar com o rosto voltado para a direção oeste. Se o recruta está inicialmente com o rosto voltado para o leste, após um comando "direita, volver!" ele deve ter o rosto voltado para o sul.
No entanto, durante o treinamento, em que o recruta tinha inicialmente o rosto voltado para o norte, o sargento emitiu uma série tão extensa de comandos, e tão rapidamente, que até ele ficou confuso, e não sabe mais para qual direção o recruta deve ter seu rosto voltado após executar todos os comandos. Você pode ajudar o sargento?
A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N que indica o número de comandos emitidos pelo sargento ($1 \leq N \leq 1.000$). A segunda linha contém N caracteres, descrevendo a série de comandos emitidos pelo sargento. Cada comando é representado por uma letra: 'E' (para "esquerda, volver!") e 'D' (para "direita, volver!"). O final da entrada é indicado por $N = 0$.
Para cada caso de teste da entrada seu programa deve produzir uma única linha da saída, indicando a direção para a qual o recruta deve ter sua face voltada após executar a série de comandos, considerando que no início o recruta tem a face voltada para o norte. A linha deve conter uma letra entre 'N', 'L', 'S' e 'O', representando respectivamente as direções norte, leste, sul e oeste.

Exemplo de entrada	**Saída para o exemplo de entrada**
3 DDE 2 EE 0	L S

3. **(bicho)** O Jogo do Bicho é um jogo baseado em números. Seu nome deriva do fato que os números são divididos em 25 grupos, dependendo do valor dos dois últimos dígitos (dezenas e unidades), e cada grupo tem associado um animal da seguinte forma: o primeiro grupo (burro) consiste nos números 01, 02, 03 e 04; o segundo grupo (águia) é composto dos números 05, 06, 07 e 08; e assim em diante, até o último grupo contendo os números 97, 98, 99 e 00.
As regras do jogo são simples. No momento da aposta, o jogador decide o valor da aposta V e um número N ($0 \leq N \leq 1000000$). Todos os dias, na praça principal da cidade, um número M é sorteado ($0 \leq M \leq 1000000$). O prêmio de cada apostador é calculado da seguinte forma:

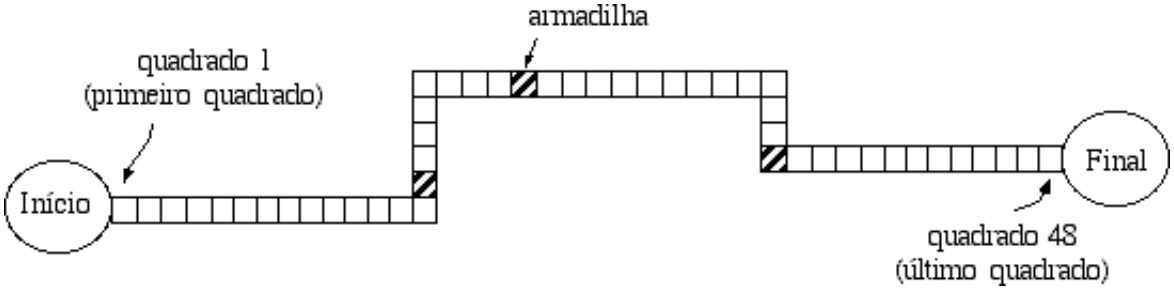
   - se M e N têm os mesmos quatro últimos dígitos (milhar, centena, dezena e unidade), o apostador recebe $V \times 3000$ (por exemplo, $N = 99301$ e $M = 19301$);
   - se M e N têm os mesmos três últimos dígitos (centena, dezena e unidade), o apostador recebe $V \times 500$ (por exemplo, $N = 38944$ e $M = 83944$);
   - se M e N têm os mesmos dois últimos dígitos (dezena e unidades), o apostador recebe $V \times 50$ (por exemplo, $N = 111$ e $M = 552211$);

- se M e N têm os dois últimos dígitos no mesmo grupo, correspondendo ao mesmo animal, o apostador recebe $V \times 16$ (por exemplo, $N = 82197$ and $M = 337600$);
- se nenhum dos casos acima ocorrer, o apostador não recebe nada.

Obviamente, o prêmio dado a cada apostador é o máximo possível de acordo com as regras acima. No entanto, não é possível acumular prêmios, de forma que apenas um dos critérios acima deve ser aplicado no cálculo do prêmio. Se um número N ou M com menos de quatro dígitos for apostado ou sorteado, assuma que dígitos 0 devem ser adicionados na frente do número para que se torne de quatro dígitos; por exemplo, 17 corresponde a 0017.
Dado o valor apostado, o número escolhido pelo apostador, e o número sorteado, faça um programa que calcule qual o prêmio que o apostador deve receber.
ENTRADA: A entrada contém vários casos de teste. Cada caso consiste em apenas uma linha, contendo um número real V e dois inteiros N e M, representando respectivamente o valor da aposta com duas casas decimais ($0.01 \leq V \leq 5000.00$), o número escolhido para a aposta ($0 \leq N \leq 1000000$) e o número sorteado ($0 \leq M \leq 1000000$). O final da entrada é indicado por uma linha contendo $V = M = N = 0$.
SAÍDA:Para cada um dos casos de teste seu programa deve imprimir uma linha contendo um número real, com duas casas decimais, representando o valor do prêmio correspondente a aposta dada.

Exemplo de entrada	**Saída para o exemplo de entrada**
32.20 32 213929	515.20
10.50 32 213032	5250.00
2000.00 340000 0	6000000.00
520.00 874675 928567	0.00
10.00 1111 578311	500.00
0 0 0	

4. **(trilha)** Um jogo simples que tem divertido gerações de crianças consiste de um tabuleiro contendo uma trilha de quadrados e um conjunto de peças coloridas. No início do jogo cada jogador recebe uma peça; todas as peças são inicialmente posicionadas imediatamente antes do primeiro quadrado da trilha.
   O jogo progride em turnos. A cada turno, jogadores jogam um par de dados, e movem suas peças para a frente. As peças são movidas sempre para a frente, pelo número de quadrados correspondente à soma dos pontos obtidos nos dados. A ordem em que os jogadores jogam os dados é sempre a mesma nos turnos (jogador A, depois jogador B, etc.).
   A maioria dos quadrados da trilha no tabuleiro são 'normais', mas alguns são 'armadilhas'. Se a peça de um jogador cai em uma armadilha ao final do movimento de um jogador, o jogador perde a vez de jogar no próximo turno. Ou seja, ele/ela não joga os dados e sua peça fica um turno sem ser movimentada.
   Há exatamente três armadilhas na trilha do tabuleiro.



O vencedor do jogo é o jogador cuja peça alcance o final da trilha primeiro. O final da trilha é após o último quadrado da trilha. Considere, por exemplo, o tabuleiro da figura acima, cujos quadrados são numerados de 1 a 48. No início do jogo, as peças estão posicionadas no local marcado 'Início' na fugura, ou seja, antes do quadrado número 1. Portanto, se um jogador obtém um resultado 7 nos dados (dados marcando 2 e 5, por exemplo), sua peça é posicionada no quadrado número 7 ao final do primeiro turno do jogo. Além do mais, se a peça de um jogador está posicionada no quadrado 41, o jogador necessita de um resultado pelo menos igual a 8 nos dados para alcançar o final da trilha e vencer o jogo. Note ainda que não há empates no jogo.

TAREFA: São fornecidos o número de jogadores, o número de quadrados na trilha, a posição das armadilhas e uma lista de resultados de dados. Você deve escrever um programa que determine o vencedor do jogo.

ENTRADA: Seu programa deve processar vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois inteiros J e Q representando respectivamente o número de jogadores e o número de quadrados na trilha ($1 \leq J \leq 10$ e $3 \leq Q \leq 10000$). A segunda linha de um conjunto de teste descreve as armadilhas, representadas por três inteiros distintos $T1$, $T2$ e $T3$, denotando as suas posições na trilha ($1 \leq T1, T2, T3 \leq Q$). A seguir é fornecido o conjunto de resultados dos dados. Cada resultado é descrito em uma linha separada, cada linha contendo inteiros $D1$ e $D2$ ($1 \leq D1, D2 \leq 6$), que representam os resultados dos dados. O número de resultados dos dados em um conjunto de teste será sempre o número exato necessário para que um jogador vença o jogo. O final da entrada é indicado por $J = Q = 0$.

Um jogador é identificado por um número de 1 a J. Jogadores jogam em ordem sequencial, de 1 a J.

SAÍDA: Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. A segunda linha deve conterum único inteiro, identificando o vencedor. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de entrada	**Saída para o exemplo de entrada**
2 10 2 4 8 1 1 3 4 1 2 6 5 3 7 4 5 7 1 2 2 2 2 1 1 1 1 2 1 1 1 1 0 0	Teste 1 1 Teste 2 3

5. **(cbd)** Um circuito bioquímico digital (CBD) é um artefato composto de um conjunto de *pontos de processamento*. Cada ponto de processamento é constituído por um minúsculo receptáculo para reagentes bioquímicos, feito de um substrato biológico que se comporta como um micro-circuito eletônico digital. Dependendo do estado da reação no receptáculo, o substrato gera dois níveis de voltagem. Um leitor acoplado ao CBD é capaz de realizar a leitura de todos os pontos de processamento de um CDB num dado instante, interpretando os dois níveis de voltagem como 0 ou 1.
   Um experimento com o CBD é realizado da seguinte maneira. Os pontos de processamento são carregados com as subst ancias de interesse e reagentes apropriados e, a cada intervalo fixo de tempo (tipicamente alguns milisegundos), os pontos de processamento são lidos. Assim, o experimento resulta em uma seqüência de conjuntos (vetores) de bits, cada vetor correspondendo a uma medição do CBD.
   Uma seqüência ininterrupta de bits 1 de um mesmo ponto de processamento ao longo do tempo é denominada de *palito*. O *comprimento* de um palito é o número de bits 1 que o compõe (note que o comprimento dos palitos de um experimento pode variar entre um e o número de medições efetuadas). Uma característica importante de um experimento com o CBD é a quantidade e o comprimento dos palitos gerados. A figura abaixo mostra o resultado de um experimento realizado com um CBD de seis pontos de processamento, em que foram efetuadas quatro medições, contendo três palitos de comprimento um, um palito de comprimento dois e um palito de comprimento quatro.

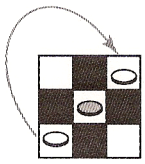
```
0 1 0 1 1 0
0 0 0 1 0 0
0 1 0 1 0 1
0 1 0 1 0 0
```

   Você foi contratado para escrever um programa que determine, dado o resultado de um experimento,

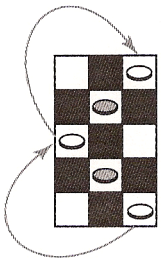
quantos palitos de comprimento igual ou maior do que um certo valor foram gerados.
ENTRADA: A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três inteiros P,N e C que indicam respectivamente o número de pontos de processamento ($1 \leq P \leq 1000$), o número de medições efetuadas ($1 \leq N \leq 1000$) e o comprimento mínimo de palitos de interesse ($1 \leq C \leq N$). Cada uma das próximas N linhas contém seqüências de P dígitos $0, 1$, separados por um espaço em branco. O final da entrada é indicado por $P = N = C = 0$.
SAÍDA: Para cada caso de teste da entrada seu programa deve produzir uma única linha da saída, contendo o número de palitos de comprimento maior ou igual a C produzidos pelo experimento.

Exemplo de entrada	**Saída para o exemplo de entrada**
2 2 2 1 1 1 1 4 5 3 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0	2 2

6. **(megadamas)** MegaDamas é um jogo de tabuleiro para dois jogadores, muito similar ao conhecido jogo de Damas. O tabuleiro é retangular, com N linhas e M colunas de pequenos quadrados arranjados em uma grade $N \times M$. Os pequenos quadrados são alternadamente coloridos com uma cor clara e uma cor escura, no padrão usual de um tabuleiro de damas. Os quadrados de cor escura são denominados casas (note no entanto que, por raz oes de visualização, os diagramas abaixo mostram casas como quadrados brancos).
   No início do jogo, cada jogador tem um certo número de peças, posicionadas nas casas mais próximas da borda do tabuleiro que o jogador escolher (os jogadores escolhem bordas opostas). Durante o jogo, as peças só podem ocupar as casas do tabuleiro.
   Um dos movimentos do jogo é capturar uma peça do oponente, saltando sobre ela, diagonalmente, para a casa adjacente além da peça, casa esta que deve estar vazia. A peça do oponente é então removida do tabuleiro. As três casas envolvidas na captura (a casa inicial de sua peça, a casa que contém a peça do oponente e a casa vazia, onde sua peça estará após a jogada) devem estar diagonalmente alinhadas e devem ser diagonalmente adjacentes, como no diagrama abaixo.



Captura simples



Captura múltipla

Em MegaDamas uma peça pode capturar peças do oponente saltando diagonalmente para a frente ou para trás (note que, na maioria das variações existentes do jogos de Damas, uma peça só pode capturar peças oponentes saltando para a frente). Você pode também efetuar uma captura múltipla, com uma peça apenas, saltando seguidamente para casas vazias sobre peças oponentes. Em uma captura múltipla, a sua peça pode mudar de direção, saltando primeiro em uma direção e depois em outra. Você pode capturar apenas uma peça a cada salto, mas pode capturar várias peças com saltos seguidos. Você não pode saltar sobre uma peça sua, e não pode saltar a mesma peça oponente mais de uma vez.

São dadas as dimens oes do tabuleiro e uma descrição do estado corrente de um jogo. é a sua vez de jogar e você deve determinar o número máximo de peças do seu oponente que podem ser capturadas em um movimento de captura.

ENTRADA: A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros N e M indicando respectivamente o número de linhas e o número de colunas do tabuleiro ($3 \leq N \leq 20$, $3 \leq M \leq 20$ e $N \times M \leq 200$). O quadrado mais à esquerda do tabuleiro na borda mais próxima ao jogador é uma casa. A segunda linha contém a descrição do estado do jogo.

Cada descrição consiste de $\lceil (N \times M)/2 \rceil$ inteiros, separados por um espaço, correspondendo às casas do tabuleiro, que são numeradas de 1 a $\lceil (N \times M)/2 \rceil$, da esquerda para a direita, da borda mais próxima ao jogador à borda mais próxima ao seu oponente. Na descrição do estado do jogo, '0' representa uma casa vazia, 1 representa uma casa com uma de suas peças, e '2' representa uma casa com uma peça de seu oponente. Há no máximo $\lfloor (N \times M)/4 \rfloor$ peças de cada jogador no tabuleiro. O final da entrada é indicado por $N = M = 0$.

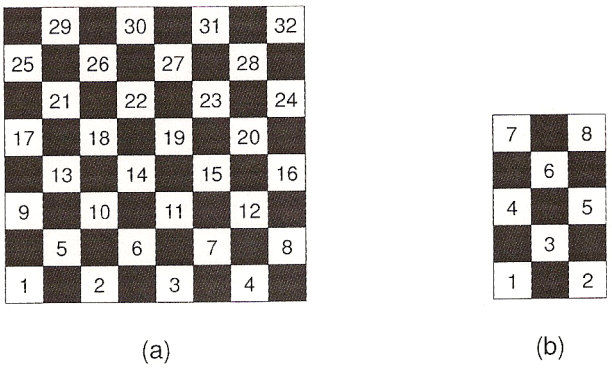


Figura 1: Numeração das casas em (a) tabuleiro de dimensões 8×8 e em (b) tabuleiro de dimensões 5×3.

A entrada deve ser lida da entrada padrão.

SAÍDA: Para cada caso de teste da entrada, seu programa deve produzir uma única linha na saída, contendo um inteiro indicando o maior número de peças de seu oponente que podem ser capturadas em uma jogada.

Exemplo de entrada
3 3 2 1 2 0 1 5 3 1 0 2 1 0 2 0 0 8 8 2 2 2 2 0 0 0 0 2 2 2 2 0 0 0 0 2 2 2 2 0 0 0 0 2 2 2 2 0 1 0 0 0 0
Saída para o exemplo de entrada
1 2 7

7. **(rotas)** Uma tragédia aconteceu recentemente em sua cidade. Um paciente em condição crítica, que necessitava tratamento urgente, morreu enquanto era transportado para um grande hospital da capital do estado. O que ocorreu foi que a ambul ancia ficou presa no trânsito, devido a uma rocha que deslizou na estrada. A população reclamou com o governador, que agora deseja evitar acontecimentos similares no futuro. Infelizmente, deslizamentos de rochas são muito comuns nesse estado, com muitas montanhas e serras. Assim, para minimizar o número de tragédias devidas a deslizamentos de rochas e outros imprevistos, o governador decidiu criar rotas alternativas entre cada cidade do estado e a capital. Para isso, é necessário inicialmente identificar quais segmentos de estradas são atualmente críticos, isto é, se bloqueados causam que não haja caminho possível entre alguma cidade e a capital. Um segmento de estrada é um trecho de estrada que liga duas cidades distintas.
   Sua tarefa é escrever um programa para identificar esses segmentos críticos de estradas. **ENTRADA:** A entrada é composta de vários casos de testes. A primeira linha de um caso de teste contém dois inteiros N e M que indicam respectivamente o número de cidades ($2 \leq N \leq 100$) e o número de segmentos de estrada ($1 \leq M \leq 10000$). Cada uma das N linhas seguintes contém o nome de uma cidade (apenas letras minúsculas e maiúsculas, comprimento máximo de 20 caracteres). A primeira dessas cidades é a capital do estado. Cada uma das M linhas seguintes descreve um segmento de estrada, contendo um par de nomes de cidades separados por um espaço em branco. Note que, como as montanhas causam dificuldade na construção de estradas, muitos segmentos de estrada são de mão única. Um segmento com duas mãos é representado por dois trechos de mão única. Você deve supor que existe ao menos um caminho de cada cidade para a capital. O final da entrada é indicado por $N = M = 0$. **SAÍDA:** Para cada caso de teste seu programa deve listar os segmentos críticos, com um segmento crítico por linha. Cada segmento crítico deve ser representado por dois nomes de cidades separados por um espaço em branco. Os segmentos críticos de estrada devem ser listados na mesma ordem em que aparecem na entrada; para cada segmento, as cidades devem ser listadas na mesma ordem em que aparecem na entrada. Se não existir nenhum segmento crítico seu programa deve imprimir uma linha contendo apenas a palavra `Nenhuma` . Imprima uma linha em branco após cada caso de teste.

Exemplo de entrada	**Saída para o exemplo de entrada**
6 10 PortoAlegre Gramado Canela NovoHamburgo Pelotas RioGrande Canela Gramado Canela NovoHamburgo Gramado NovoHamburgo NovoHamburgo PortoAlegre PortoAlegre NovoHamburgo RioGrande Pelotas Pelotas PortoAlegre PortoAlegre Pelotas Pelotas RioGrande NovoHamburgo Canela 3 5 Sacramento SanFrancisco SantaClara SanFrancisco Sacramento Sacramento SantaClara SantaClara SanFrancisco SanFrancisco Sacramento Sacramento SanFrancisco 3 4 Recife Olinda Paulista Olinda Recife Paulista Recife Olinda Paulista Paulista Olinda 0 0	Gramado NovoHamburgo NovoHamburgo PortoAlegre RioGrande Pelotas Pelotas PortoAlegre SantaClara SanFrancisco Nenhuma

8. **(batalha)** Um determinado exército numa certa fronteira decidiu enumerar as coordenadas em sua volta de maneira a tornar difícil para o inimigo saber a quais posições eles estão se referindo no caso de o sinal de rádio usado para comunicação ser interceptado. O processo de enumeração escolhido foi o seguinte: primeiro decide-se onde ficam os eixos `x` e `y`; a seguir, define-se uma equação linear que descreva a posição da fronteira em relação aos eixos (sim, ela é uma linha reta); finalmente, enumeram-se todos os pontos do plano cartesiano que não fazem parte da fronteira, sendo o número 0 atribuído à coordenada `(0,0)` e daí em diante atribuindo-se os números para as coordenadas inteiras seguindo uma espiral de sentido horário, sempre pulando os pontos que caem em cima da fronteira (veja a Figura 1). Caso o ponto `(0,0)` caia em cima da fronteira, o número `0` é atribuído ao primeiro ponto que não faça parte da fronteira seguindo a ordem especificada.

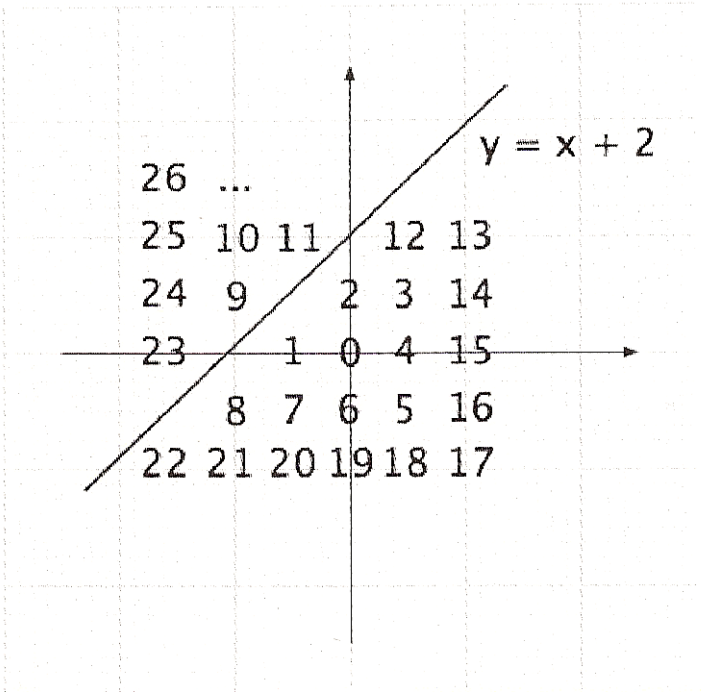


Figura 2: Enumeração dos pontos das coordenadas inteiras

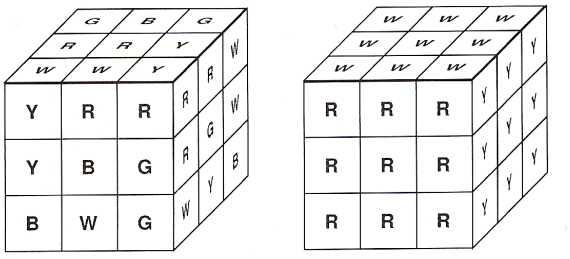
De fato o inimigo não tem como saber a qual posição o exército se refere, a não ser que o inimigo saiba o sistema usado para enumerar os pontos. Tal esquema, porém, complicou a vida do exército, uma vez que é difícil determinar se dois pontos quaisquer estão no mesmo lado da fronteira ou em lados opostos. é aí que eles precisam da sua ajuda.

ENTRADA: A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 100$) que representa a quantidade de casos de teste. Seguem-se os N casos de teste. A primeira linha de cada caso de teste contém dois inteiros a e b ($-5 \leq a \leq 5$ e $-10 \leq b \leq 10$), que descrevem a equação da fronteira: $y = ax + b$. A segunda linha de cada caso de teste contém um inteiro K, indicando a quantidade de consultas que se seguem ($1 \leq K \leq 1000$). Cada uma das K linhas seguintes descreve uma consulta, sendo composta por dois inteiros M e N representando as coordenadas enumeradas de dois pontos ($0 \leq M, N \leq 65535$).

SAÍDA: Para cada caso de teste da entrada seu programa deve produzir $K + 1$ linhas. A primeira linha deve conter a identificação do caso de teste na forma `Caso X`, onde `X` deve ser substituído pelo número do caso (iniciando de 1). As K seguintes devem conter os resultados das K consultas feitas no caso correspondente da entrada, na forma: `Mesmo lado da fronteira` ou `Lados opostos da fronteira`.

Exemplo de entrada	**Saída para o exemplo de entrada**
2	Caso 1
1 2	Mesmo lado da fronteira
10	Mesmo lado da fronteira
26 25	Mesmo lado da fronteira
25 11	Mesmo lado da fronteira
24 9	Mesmo lado da fronteira
23 28	Lados opostos da fronteira
25 9	Lados opostos da fronteira
25 1	Lados opostos da fronteira
25 0	Lados opostos da fronteira
9 1	Lados opostos da fronteira
23 12	Caso 2
26 17	Mesmo lado da fronteira
1 2	Mesmo lado da fronteira
12	Mesmo lado da fronteira
0 1	Mesmo lado da fronteira
1 2	Mesmo lado da fronteira
2 3	Mesmo lado da fronteira
3 4	Mesmo lado da fronteira
4 5	Mesmo lado da fronteira
5 6	Lados opostos da fronteira
6 7	Mesmo lado da fronteira
7 8	Mesmo lado da fronteira
8 9	Lados opostos da fronteira
9 10	
10 11	
11 12	

9. **(cubomagico)** Um brinquedinho muito conhecido, chamado Cubo Mágico, consiste de um cubo como mostrado na Figura 3a, onde as letras representam cores (e.g. B para azul, R para vermelho, ...). O objetivo do jogo é rotacionar as faces do cubo de tal forma que ao final cada face tem uma cor diferente, como mostra a Figura 3b. Note que, quando uma face é rotacionada, a configuração de cores nas faces adjacentes muda.



(a) Misturado (b) Posição Vencedora

Figura 3: Cubo Mágico

A Figura 4 ilustra uma rotação de uma das faces.

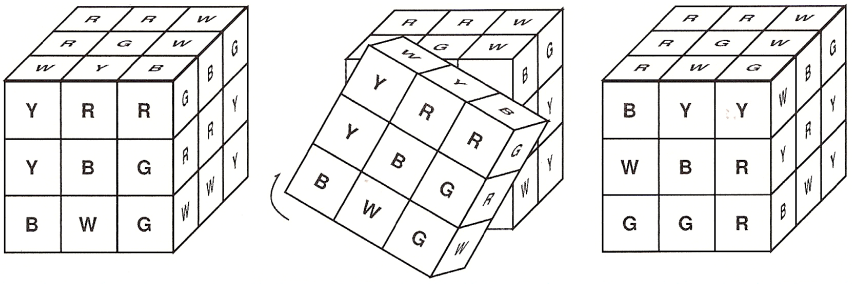
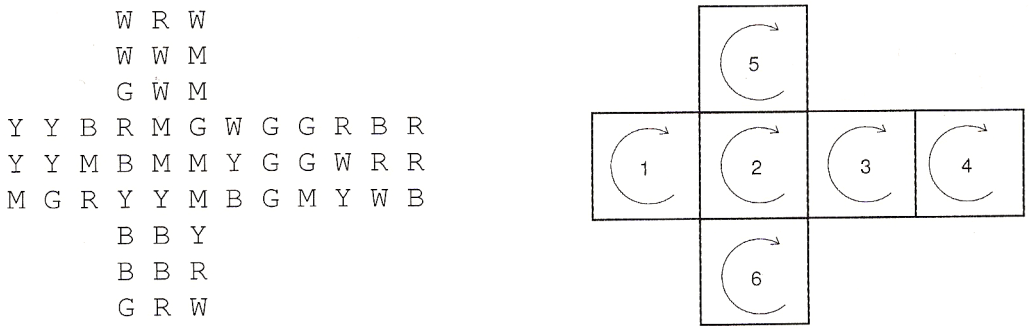


Figura 4: Exemplo de rotação

Outro dia seu avô, que tem muitos anos de experiência, afirmou que, a partir de qualquer configuração inicial do Cubo Mágico, ele era capaz de indicar uma sequência de rotações até chegar a uma configuração vencedora. Para conferir se ele é bom mesmo, você precisa fazer um programa que verifique se as rotações que seu avô indica dão resultado ou não.
As faces do cubo são representadas conforme a Figura 5a. As seis cores são Amarelo (Y), Vermelho (R), Azul (B), Verde (G), Branco (W) e Magenta (M). O jogador (no caso, seu avô), informa uma configuração inicial e uma lista de rotações. A rotação é representada por um número inteiro, indicando a face a ser rotacionada e a direção da rotação (valores positivos indicam sentido horário, e valores negativos indicam sentido anti-horário). As faces do cubo são numeradas conforme a Figura 5b.



(a) Cores (b) Ident. de faces para rotação

Figura 5: Representação do cubo

ENTRADA: A entrada contém vários casos de teste. A primeira linha da entrada contém um valor inteiro que indica o número de testes. Cada descrição de teste consiste de 10 linhas. As primeiras 9 linhas descrevem a configuração inicial, no formato mostrado na Figura 5a. A linha seguinte contém uma lista de rotações, terminando com o valor 0.
SAÍDA: Para cada caso de teste seu programa deve imprimir uma mensagem. Se seu avô está correto, a mensagem é `Valeu, véio!`, caso contrário a mensagem deve ser `Errou, reboco de igreja véia !!!`.

Exemplo de entrada	**Saída para o exemplo de entrada**
3 G Y Y G Y Y G Y Y W W WY R R M M MG G B W W WY R R M M MG G B W W WY R R M M MG G B R B B R B B R B B -1 0 G Y Y G Y Y G Y Y W W WY R R M M MG G B W M WY R R M W MG G B W W WY R R M M MG G B R B B R B B R B B -1 0 M W M W W G W W Y G Y Y M M B M B G W R B B Y Y M M B M G G W R R Y M G W B B R R G R R W R Y Y G B Y R G B +4 +6 -2 +3 -4 +2 -3 -6 0	Valeu, véio! Errou, reboco de igreja véia !!! Valeu, véio!

10. **(ficha)** Scanners baratos podem apenas adquirir imagens em tons de cinza, onde cada ponto (pixel) é representado por um valor inteiro na faixa [0..255]. Uma companhia que fabrica máquinas de venda automáticas quer usar estes scanners para validar as fichas usadas em suas máquinas. As fichas são pequenas peças quadradas de metal com buracos estrategicamente colocados. Fichas com diferentes buracos são usadas para valores diferentes.

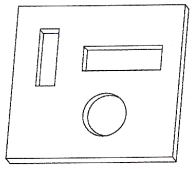


Figura 6: Ficha da máquina de vender

Um scanner produz uma imagem da ficha introduzida pelo cliente e um programa de computador valida a ficha. Na imagem gerada pelo scanner, o metal aparece como pixels escuros (valores próximos de 0) e buracos aparecem como pixels mais claros (valores próximos de 255).
Existem dois problemas que devem ser resolvidos no processo de validação. O primeiro problema é que, uma vez que a ficha é quadrada, um cliente pode introduzi-la na máquina em diversas posições. O segundo problema é devido à qualidade ruim da imagem gerada pelos scanners das máquinas. Para validar a ficha, a máquina deve comparar a saída do scanner com uma imagem padrão do token, produzida previamente usando um scanner de alta qualidade.
Você deve escrever um programa que, dada uma imagem padrão de uma ficha e uma imagem produzida pelo scanner da máquina, determine o grau de confiança da ficha introduzida. O grau de confiança é a percentagem de pixels na imagem do scanner da máquina cuja intensidade difere de 100 ou menos do pixel correspondente na imagem padrão. Como uma ficha pode ser introduzida na máquinaa de diversas formas diferentes, estamos interessados no grau de confiança mais alto, considerando todas as posições possíveis da ficha.
ENTRADA: Seu programa deve processar diversos casos de teste. Cada caso especifica o tamanho em pixels da ficha e os valores dos pixels para a imagem padrão e para a imagem escaneada. A primeira linha de um caso de teste contém um inteiro L que indica o tamanho da imagem em pixels ($1 \leq L \leq 400$). As L linhas seguintes contém L inteiros cada, representando os valores dos pixels para a imagem padrão. Em seguida, as próximas L linhas contém os valores dos pixels da imagem escaneada. O final da entrada é indicado por $L = 0$.
SAÍDA: Para cada caso de teste seu programa deve imprimir uma única mensagem contendo o grau de confiança da imagem correspondente. O grau de confiança deve ser impresso como um número real com precisão de 2 dígitos decimais, e o último dígito decimal deve ser arredondado.

Exemplo de entrada	Saída para o exemplo de entrada
4 250 251 249 250 251 120 245 248 248 5 190 247 5 5 180 246 0 1 240 240 250 2 250 254 244 251 255 253 230 250 250 252 3 250 250 250 150 0 150 250 2 250 253 150 253 0 2 248 251 150 250 5 255 255 255 255 255 255 0 255 0 0 255 0 0 255 255 255 255 0 255 255 255 255 255 255 0 255 0 255 255 0 255 0 255 255 255 255 255 0 0 255 255 0 0 255 255 154 154 255 255 255 0	93.75 100.00 92.00