Maratona de Programação DCC/UFJF – 2014

15/10/2014

Caderno de Problemas

- Este caderno contém 6 (seis) questões -

Instruções

1) O time que conseguir resolver o maior número de problemas (no menor tempo acumulado com as penalidades, caso haja empate) é declarado o vencedor;
2) Durante a Maratona de Programação NÃO É PERMITIDO nenhum contato direto com a organização, técnicos ou juízes. Caso seja necessário qualquer contato, formule uma pergunta (*clarification*) no próprio ambiente da competição;
3) É permitida a consulta de material impresso, tais como livros ou listagens. Porém, não será permitido o acesso à Internet ou a qualquer material digital, tais como CDs ou pendrives;
4) É totalmente vedada a comunicação com membros de outras equipes ou com qualquer outra pessoa que não seja membro da sua equipe;
5) Em todos os problemas, as entradas e saídas de dados devem ser feitas por meio da entrada e saída padrões;
6) O ambiente computacional utilizado nesta competição é o ambiente oficial da Maratona de Programação, assim a correção das soluções propostas será feita via ambiente, não sendo necessário "entregar" as soluções aos juízes. Apenas submeta a solução;
7) Os problemas não estão ordenados por ordem de dificuldade ou por qualquer outra ordem;
8) Utilize para nomear os arquivos os nomes indicados nos enunciados dos problemas. Por exemplo, se o nome indicado for ***teste***, o nome do arquivo com a solução deverá ser: ***teste.c*** ou ***teste.cpp***.

Problema A

Aritmética com Morse

Arquivo: `morse.[c|cpp]`

O código Morse é um método usado para transmitir mensagens de texto como uma série de pontos "." e traços "-". Por exemplo, a letra "A" é representada por ".-" e a letra "B" com "-...". Este código tem sido utilizado por vários anos em aplicações militares e civis, mas você vai usá-lo para fazer matemática. Com isto em mente, atribuímos valores aos pontos e traços e, para economizar espaço, usamos dois caracteres adicionais. A tabela a seguir mostra os quatro caracteres permitidos e seus valores.

Caractere	Valor	
.	1	
-	5	
:	2	(duas vezes ".")
=	10	(duas vezes "-")

Um número Morse é uma cadeia que contém apenas os quatro caracteres acima; seu valor é a soma dos valores atribuídos a cada caractere individualmente. Como exemplo, o valor de "=.-.." é 10 + 1 + 5 + 1 + 1 = 18. Note que cada número Morse representa um único valor, mas existem valores que podem ser representados com diversos números Morse. Por exemplo, existem três números Morse com valor 3: "...", ".:" e ":.".

Bem, os números estão prontos. Para formar expressões precisamos também de operadores. Consideramos dois operadores aritméticos: "+" representa adição, enquanto "*" representa multiplicação. Uma expressão Morse é uma sequência de cadeias que intercala os números e operadores Morse, que começa e termina com um número Morse e contém pelo menos um operador. Expressões Morse podem ser avaliadas através da substituição de cada número Morse pelo seu valor e, em seguida, da avaliação da expressão matemática obtida utilizando a precedência e a associatividade padrões dos operadores. Por exemplo, o valor da expressão Morse "=.-.. + ... * :." é 18 + 3 x 3 = 18 + (3 x 3) = 27. Dada a expressão Morse, imprima seu valor.

Entrada

A entrada é composta por diversos casos de teste. A primeira linha contém um inteiro *N* representando o número de operadores na expressão Morse. A segunda linha contém 2*N* + 1 cadeias não vazias representando a expressão Morse. A linha intercala os números e operadores Morse, sendo a primeira e a última cadeias números Morse. Cada número Morse tem no máximo 7 caracteres, onde cada caractere é "-", ".", ":" ou "=". Cada operador é "+" ou "*". A entrada é terminada por um conjunto onde *N* = 0. Este conjunto não deve ser processado.

Saída

Para cada um dos casos de teste seu programa deve imprimir uma única linha contendo um número inteiro, representando o valor da expressão Morse.

Restrições

- $1 \leq N \leq 4$

Exemplos

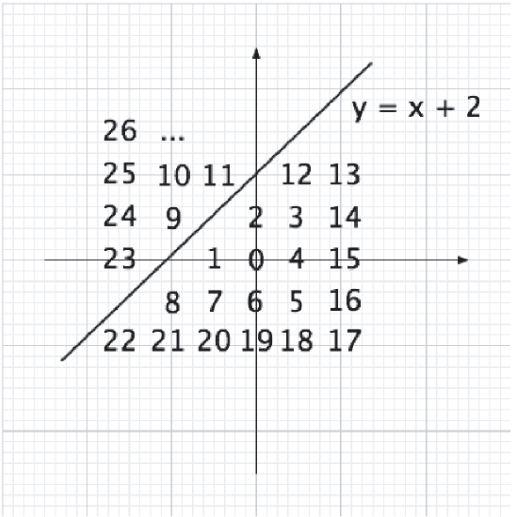
Exemplo de entrada	Saída para o exemplo de entrada
2 =.-.. + ... * :. 3 - * - + - * - 0	27 50

Problema B

Amigos ou Inimigos?

Arquivo: `amigos.[c|cpp]`

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 abaixo). 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.



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* 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*, 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. Cada uma das *K* linhas seguintes descreve uma consulta, sendo composta por dois inteiros *X* e *Y* representando as coordenadas enumeradas de dois pontos. A entrada termina quando *N* = 0.

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
```

Restrições

- $1 \leq N \leq 100$
- $-5 \leq a \leq 5$ e $-10 \leq b \leq 10$
- $1 \leq K \leq 1000$
- $0 \leq X, Y \leq 65535$

Exemplos

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

Problema C

Mergulho

Arquivo: `mergulho.[c|cpp]`

O recente terremoto em Nlogônia não chegou a afetar muito as edificações da capital, principal epicentro do abalo. Mas os cientistas detectaram que o principal dique de contenção teve um dano significativo na sua parte subterrânea que, se não for consertado rapidamente, pode causar o seu desmoronamento, com a consequente inundação de toda a capital.

O conserto deve ser feito por mergulhadores, a uma grande profundidade, em condições extremamente difíceis e perigosas. Mas como é a sobrevivência da própria cidade que está em jogo, seus moradores acudiram, em grande número, como voluntários para essa perigosa missão.

Como é tradicional em missões perigosas, cada mergulhador recebeu no início do mergulho uma pequena placa com um número de identificação. Ao terminar o mergulho, os voluntários devolviam a placa de identificação, colocando-a em um repositório.

O dique voltou a ser seguro, mas aparentemente alguns voluntários não voltaram do mergulho. Você foi contratado para a penosa tarefa de, dadas as placas colocadas no repositório, determinar quais voluntários perderam a vida salvando a cidade.

Entrada

A entrada contém diversos casos de testes. Cada caso de teste é composto por duas linhas. A primeira linha contém dois inteiros N e R, indicando respectivamente o número de voluntários que mergulhou e o número de voluntários que retornou do mergulho. Os voluntários são identificados por números de 1 a N. A segunda linha de cada caso de teste contém R inteiros, indicando os voluntários que retornaram do mergulho (ao menos um voluntário retorna do mergulho). A leitura dos valores $N = R = 0$ encerra a leitura do programa e não deve ser processada.

Saída

Seu programa deve produzir uma única linha, contendo os identificadores dos voluntários que não retornaram do mergulho, na ordem crescente de suas identificações. Deixe um espaço em branco após cada identificador (note que isto significa que deve haver um espaço em branco também após o último identificador). Se todos os voluntários retornaram do mergulho, imprima apenas o caractere '*' (asterisco).

Restrições

- $1 \leq R \leq N \leq 10^4$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada

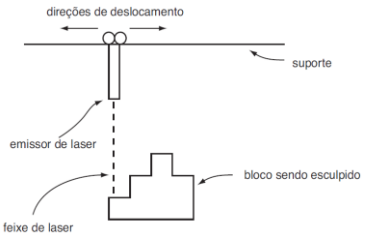
`5 3`
`3 1 5`
`6 6`
`6 1 3 2 5 4`
`0 0`
	`2 4`
	`*`

Problema D

Escultura a Laser

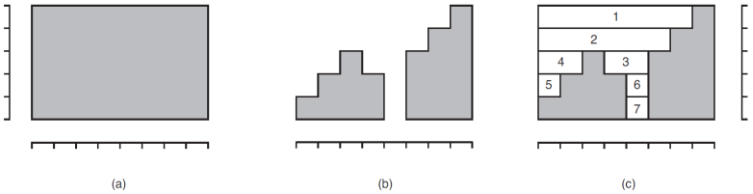
Arquivo: `laser.[c|cpp]`

Desde a sua invenção, em 1958, os raios laser têm sido utilizados em uma imensa variedade de aplicações, como equipamentos eletrônicos, instrumentos cirúrgicos, armamentos, e muito mais.



A figura ao lado mostra um diagrama esquemático de um equipamento para esculpir, com laser, um bloco de material maciço. Na figura vemos um emissor laser que se desloca horizontalmente para a direita e para a esquerda com velocidade constante. Quando o emissor é ligado durante o deslocamento, uma camada de espessura constante é removida do bloco, sendo vaporizada pelo laser.

A figura abaixo ilustra o processo de escultura a laser, mostrando um exemplo de (a) um bloco, com 5 mm de altura por 8 mm de comprimento, no início do processo, (b) o formato que se deseja que o bloco esculpido tenha, e (c) a sequência de remoção das camadas do bloco durante o processo, considerando que a cada varredura uma camada de espessura de 1 mm é removida. Na primeira varredura, o pedaço numerado como 1 é removido; na segunda varredura, o pedaço numerado como 2 é removido, e assim por diante. Durante o processo de remoção, o laser foi ligado um total de 7 vezes, uma vez para cada pedaço de bloco removido.



Escreva um programa que, dados a altura do bloco, o comprimento do bloco, e a forma final que o bloco deve ter, determine o número total vezes de que o laser deve ser ligado para esculpir o bloco.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por duas linhas. A primeira linha de um caso de teste contém dois números inteiros A e C, separados por um espaço em branco, indicando respectivamente a altura A e o comprimento C do bloco a ser esculpido, em milímetros. A segunda linha contém C números inteiros X_i, cada um indicando a altura final, em milímetros, do bloco entre as posições i e $i + 1$ ao longo do comprimento. Considere que a cada varredura uma camada de espessura 1 milímetro é removida do bloco ao longo dos pontos onde o laser está ligado. 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, contendo um número inteiro, indicando o número de vezes que o laser deve ser ligado para esculpir o bloco na forma indicada.

Restrições

- $1 \le A \le 10^4$
- $1 \le C \le 10^4$
- $0 \le X_i \le$ A, para $0 \le i \le C-1$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
5 8 1 2 3 2 0 3 4 5 3 3 1 0 2 4 3 4 4 1 0 0	7 3 3

Problema E

Botas Perdidas

Arquivo: `botas.[c|cpp]`

A divisão de Suprimentos de Botas e Calçados do Exército comprou um grande número de pares de botas de vários tamanhos para seus soldados. No entanto, por uma falha de empacotamento da fábrica contratada, nem todas as caixas entregues continham um par de botas correto, com duas botas do mesmo tamanho, uma para cada pé. O sargento mandou que os recrutas retirassem todas as botas de todas as caixas para reembalá-las, desta vez corretamente.

Quando o sargento descobriu que você sabia programar, ele solicitou com a gentileza habitual que você escrevesse um programa que, dada a lista contendo a descrição de cada bota entregue, determina quantos pares corretos de botas poderão ser formados no total.

Entrada

A primeira linha de um caso de teste contém um inteiro *N* indicando o número de botas individuais entregues. Cada uma das *N* linhas seguintes descreve uma bota, contendo um número inteiro *M* e uma letra *L*, separados por um espaço em branco. *M* indica o número do tamanho da bota e *L* indica o pé da bota: *L* = 'D' indica que a bota é para o pé direito, *L* = 'E' indica que a bota é para o pé esquerdo. O fim da entrada é indicado por uma linha onde *N* = 0, a qual não deve ser processada

Saída

Para cada caso de teste imprima uma linha contendo um único número inteiro indicando o número total de pares corretos de botas que podem ser formados.

Restrições

- $2 \leq N \leq 10^4$
- *N* é par
- $30 \leq M \leq 60$
- $L \in \{D, E\}$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 40 D 41 E 41 D 40 E 6 38 E 39 E 40 D 38 D 40 D 37 E 0	2 1

Problema F

Linhas de Contêineres

Arquivo: `linhas.[c|cpp]`

Um carregamento de Nlogs, principal produto de exportação de Nlogônia, está no porto, em contêineres, pronto para ser embarcado. Todos os contêineres têm as mesmas dimensões e são cubos. Os contêineres estão organizados no pátio do porto em *L* linhas e *C* colunas, num total de *LC* contêineres. Cada contêiner está marcado com um número de identificação distinto, de 1 a *LC*.

Cada uma das *L* linhas de contêineres será embarcada em um navio distinto. Para facilitar o desembarque nos diversos países em que serão entregues, os contêineres de uma linha devem estar organizados de forma que os números de identificação estejam ordenados. Mais precisamente, a linha 1 foi organizada no pátio de forma a conter os contêineres identificados de 1 a C ordenados crescentemente, a linha 2 de forma a conter os contêineres de *C* + 1 a 2*C* (ordenados crescentemente), e assim por diante, até a linha *L*, organizada de forma a conter os contêineres de (*L* - 1)*C* + 1 a *LC* (ordenados crescentemente). A figura (a) abaixo mostra a organização de um carregamento com 5 linhas e 4 colunas de contêineres.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

(a)

13	14	15	16
5	6	7	8
9	10	11	12
1	2	3	4
17	18	19	20

(b)

13	15	14	16
5	7	6	8
9	11	10	12
1	3	2	4
17	19	18	20

(c)

O guindaste de embarque é capaz de movimentar ou uma linha completa ou uma coluna completa de contêineres, não sendo capaz de movimentar outros tipos de agrupamentos ou contêineres individuais.

Na noite anterior ao embarque, um grupo de estivadores operou os guindastes para trocar linhas e colunas do carregamento, como forma de protestar quanto aos baixos salários. A figura (b) acima mostra a configuração dos contêineres após a troca das linhas 1 e 4; a figura (c) mostra a configuração após mais uma troca, entre as colunas 2 e 3.

O carregamento precisa ser embarcado ainda hoje, mas antes disso é necessário que os contêineres sejam reorganizados da forma descrita. Você deve escrever um programa que, dada a informação sobre a posição de cada contêiner após o protesto, determine se é possível recolocar os contêineres na forma originalmente prevista utilizando apenas os guindastes e, nesse caso, calcular o menor número de trocas de linhas e colunas necessário para esse fim.

Entrada

A primeira linha de cada caso de teste contém dois inteiros *L* e *C* indicando respectivamente o número de linhas e o número de colunas do carregamento. As *L* linhas seguintes descrevem a posição dos contêineres depois do protesto dos estivadores. Cada uma dessas *L* linhas contém *C* números inteiros $X_{l,c}$ indicando a posição de um contêiner. Cada número inteiro entre 1 e *LC* aparece na entrada, em alguma das *L* linhas. É garantido que cada número na configuração apareça uma única vez cada e que todos os números entre 1 e *LC* aparecerão na mesma. O fim da entrada é indicado por uma linha onde *L* = *C* = 0, a qual não deve ser processada.

Saída

Para cada caso de teste, seu programa deve produzir uma única linha, contendo um único inteiro, o número mínimo de trocas de linhas e colunas que devem ser realizadas pelo guindaste para recolocar os contêineres na posição original. Se não for possível colocar os contêineres na posição original, utilizando apenas trocas de linhas e colunas, imprima o caractere '*'.

Restrições

- $1 \leq L \leq 300$
- $1 \leq C \leq 300$
- $1 \leq X_{l,c} \leq LC$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 2 3 4 1 2 3 3 9 2 4 5 8 7 6 1 3 5 4 13 15 14 16 5 7 6 8 9 11 10 12 1 3 2 4 17 19 18 20 0 0	1 * 2

Warm Up

Brincadeira

Arquivo: `brinca.[c|cpp]`

Alice e Beto são amigos desde crianças. Hoje em dia estão estudando na universidade, mas sempre que se encontram relembram os tempos de infância tirando par-ou-ímpar para decidir quem escolhe o filme a ser assistido, ou qual o restaurante em que vão almoçar etc.

Ontem Alice confidenciou a Beto que ela guarda os resultados de cada vez que tiraram par-ou-ímpar desde que a brincadeira começou, no jardim de infância. Foi uma grande surpresa para Beto! Como Beto faz graduação na área de Computação, ele decidiu mostrar a Alice sua habilidade em programação, escrevendo um programa para determinar quantas vezes cada um ganhou o par-ou-ímpar no período de todos esses anos.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um único inteiro N que indica o número de vezes que os amigos tiraram par-ou-ímpar. A segunda linha de um caso de teste contém N inteiros R_i, separados por espaço, descrevendo a lista de resultados. Se $R_i = 0$ significa que Alice ganhou o i-ésimo jogo, se $R_i = 1$ significa que Beto ganhou o i-ésimo jogo. O final da entrada é indicado por $N = 0$.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída, no formato 'Alice ganhou X e Beto ganhou Y', onde $X \geq 0$ e $Y \geq 0$.

Restrições

- $1 \leq N \leq 10^4$
- $1 \leq i \leq N$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada

`5`
`0 0 1 0 1`
`6`
`0 0 0 0 0 1`
`0`
	`Alice ganhou 3 e Beto ganhou 2`
	`Alice ganhou 5 e Beto ganhou 1`

Warm Up

Quadrado

Arquivo: `quadrado.[c|cpp]`

Esse é um exemplo de como resolver problemas da Maratona de Programação. Nesse exemplo você deve encontrar o quadrado de um número.

Dica: para armazenar números inteiros grandes, utilize os modificadores de tipo *long* (longo – mais bits para armazenar), que pode ser utilizado 2 vezes (*long long*), e/ou *unsigned* (sem sinal – dobra a capacidade por não armazenar valores negativos). Para a linguagem C, utilizar o código de formatação "%llu".

Entrada

A entrada é composta por diversos casos de teste. Cada linha de um caso de teste contém um número inteiro *N*. O fim da entrada é indicado por uma linha onde *N* = 0, a qual não deve ser processada.

Saída

Para cada um dos casos de teste seu programa deve imprimir uma única linha contendo um número inteiro, representando o valor de N^2.

Restrições

- $1 \leq N \leq 10^9$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
1 2 5 10 1000000000 0	1 4 25 100 1000000000000000000

Solução

```c
#include <stdio.h>

int main()
{
    unsigned long long int n;
    scanf("%llu", &n);
    while (n != 0)
    {
        printf("%llu\n", n*n);
        scanf("%llu", &n);
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
    unsigned long long int n;
    cin >> n;
    while (n != 0)
    {
        cout << n*n << endl;
        cin >> n;
    }
    return 0;
}
```