CADERNO DE PROBLEMAS

EXEMPLO

Equipe da Robótica CPS

Abril, 2018

Instruções

1) Este caderno contém 4 problemas. As páginas estão numeradas de 1 a 11, não contando a página de rosto. Verifique se o caderno está completo antes de começar;

2) Em todos os problemas, os programas deverão ler um arquivo texto de entrada com extensão “.in” e salvar um arquivo texto de saída com a extensão “.out”, com os nomes dos arquivos indicados em cada problema;

3) Em todos os problemas, em que o final da entrada não esteja especificado no texto do problema, deverá ser considerado o final de arquivo;

4) Não é permitido em nenhuma hipótese a utilização de código pronto. Os códigos poderão ser consultados de anotações e publicações em geral, desde escritas ou impressas, e deverão ser digitados na hora da competição;

5) É terminantemente proibido a utilização da Internet para buscar códigos prontos ou qualquer outra informação, que não seja acesso e utilização da plataforma oficial da competição, isto é, o software BOCA;

6) Também não será permitido o uso de pendrives, hds externos, telefones celulares, tablets, ou qualquer outro dispositivo eletrônico, que não o único computador disponibilizado ao time;

7) Qualquer desrespeito às normas de restrições acima, o time será desclassificado;

8) Qualquer dúvida contate o pessoal do staff.

Dicas para Iniciantes

Aqui vão algumas dicas rápidas para quem não está acostumado com os tipos de problemas da maratona e de olimpíadas de informática em geral:

- Todo problema tem uma entrada exemplo e a saída esperada para aquela entrada. Você deve sempre se lembrar que essa entrada não é nem de perto a entrada que os juízes irão usar para testar seu programa. Lembre-se sempre de que é apenas um exemplo. Após fazer o programa funcionar para os exemplos, teste sempre condições extremas e de contorno (verifique os limites do problema, qual os valores mínimo e máximo que as variáveis podem atingir, etc.);

- Evite mandar programas precipitadamente. É bem melhor gastar 5 minutos testando o programa para ter certeza de que funciona para as mais diversas entradas do que receber uma mensagem de erro dos juízes (que acarreta uma penalização de 20 minutos);

- A maioria dos problemas sempre tem um "truque" escondido. Por exemplo, suponha um problema trivial de calcular fatorial. Você se lembrou de tratar o problema quando a entrada é zero? Sempre procure pensar no que o juiz deveria estar pensando para fazer a entrada. Afinal, o seu programa deve funcionar corretamente para qualquer entrada;

- Se o seu algoritmo lidar com busca, procure cortar ao máximo. Quanto menos nós expandidos, mais eficiente seu programa. Problemas de busca são, em geral, críticos com relação ao tempo, e um descuido pode acarretar em estouro do tempo limite!;

- A maratona é um torneio de criação e implementação de algoritmos. Isto significa que você nunca vai precisar se preocupar com coisas do tipo interface com o usuário, reusabilidade do código, etc. Aliás, é bem melhor usar nomes sugestivos de variáveis do que comentários;

- Evite usar as ferramentas de debug, elas consomem muito tempo. Aliás, deve-se evitar “debugar” o problema no computador, afinal, são três pessoas para apenas um computador! Enquanto você estiver “debugando” o programa na tela do computador, outros componentes do time podem estar querendo digitar um programa! O ideal é imprimir o código fonte (isso é permitido na maratona) e tentar analisar o código em busca de erros. Se for indispensável o “debug” em "tempo real" procure fazê-lo usando “print” de variáveis e técnicas do tipo. Às vezes breakpoints também podem ser úteis;

- Ninguém vai analisar seu código, portanto não interessa se ele está elegante ou não. Se você tiver uma solução “feia”, porém eficiente, essa é a que se deve usar;

- Nem sempre um algoritmo polinomial pode ser viável. Suponha que você tenha implementado um algoritmo O (n^3) para um determinado problema. Mas e se n puder assumir valores até, digamos, 1000? Com quase toda certeza seu problema

irá estourar o limite de tempo. Por isso, sempre preste atenção não só à complexidade do seu algoritmo, como à viabilidade de sua implementação;

- Não há nada pior do que gastar muito tempo fazendo e implementando um algoritmo para, depois, descobrir que ele está errado. Numa maratona, esse erro é fatal. Por isso, apesar de a pressa ser necessária, procure sempre certificar-se da corretude do algoritmo ANTES de implementá-lo!

Problema A

Fechem as portas!

Nome do arquivo fonte: portas.java, portas.php ou portas.py

Madame Beauvoir possui uma mansão onde ela recebe todos os seus descendentes (netos e bisnetos) durante as férias. Sua mansão possui exatamente N quartos (cada quarto é numerado de1 a N), onde N é também a quantidade de netos e bisnetos (cada descendente é também numerado de 1 a N).

Como toda criança, os descendentes de Mme. Beauvoir são bastante travessos. Todo dia é a mesma confusão: eles acordam de manhã cedo antes dela e se encontram no grande jardim. Cada descendente, um de cada vez, entra na mansão e troca o estado das portas dos quartos cujos números são múltiplos do seu identificador. Trocar o estado de uma porta significa fechar uma porta que estava aberta ou abrir uma porta que estava fechada. Por exemplo, o descendente cujo identificador é igual a 15 vai trocar o estado das portas 15, 30, 45, etc.

Considerando que todas as portas estão inicialmente fechadas (todos os descendentes fecham as portas antes de descer para o jardim) e que cada descendente entra exatamente uma vez na mansão (a confusão é tão grande que não sabemos em que ordem), quais portas estarão abertas após a entrada de todos os descendentes na mansão?

Entrada

A entrada contém vários casos de teste. Cada caso de teste consiste em uma linha que contém um inteiro N (0 <= N <= 25.000.000), indicando o número de portas e descendentes. O final da entrada é indicado por N = 0.

A entrada deverá ser lida de um arquivo texto chamado portas.in.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída, contendo a sequência crescente de números correspondente aos identificadores dos quartos cujas portas estarão abertas. Ao imprimir a sequência, deixe um espaço em branco entre dois elementos consecutivos.

A saída deverá ser escrita em um arquivo texto chamado portas.out.

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

Administração Central
Departamento

Problema B
Esquerda, Volver!

Nome do arquivo fonte: esquerda.java, esquerda.php ou esquerda.py

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 a 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 a 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 como 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?

Entrada

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 <= N < = 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.

A entrada deverá ser lida de um arquivo texto chamado esquerda.in.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha da saida, 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.

A saída deverá ser escrita em um arquivo texto chamado esquerda.out.

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

Problema C

Suco de Acerola

Nome do arquivo fonte: suco.java, suco.php ou suco.py

Natural das Antilhas, a acerola (Malpighia glabra Linn, também conhecida como cereja das Antilhas) já era apreciada pelos nativos das Américas há muitos séculos. Mas o grande interesse por essa fruta surgiu na década de 1940, quando cientistas porto-riquenhos descobriram que a acerola contém grande quantidade de Ácido ascórbico (vitamina C). A acerola apresenta, em uma mesma quantidade de polpa, até 100 vezes mais vitamina C do que a laranja e o limão, 20 vezes mais do que a goiaba e 10 vezes mais do que o caju e a amora.

Um grupo de amigos está visitando o Sítio do Picapau Amarelo, renomado produtor de acerola. Com a permissão de Dona Benta, dona do sítio, colheram uma boa quantidade de frutas, e pretendem agora fazer suco de acerola, que será dividido igualmente entre os amigos durante o lanche da tarde.

Conhecendo o número de amigos, a quantidade de frutas colhidas, e sabendo que cada unidade da fruta é suficiente para produzir 50 ml de suco, escreva um programa para determinar qual o volume, em litros, que cada amigo poderá tomar.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por uma única linha, contendo dois números inteiros N e F, indicando respectivamente o número de amigos ($1 <= N <= 10^3$) e a quantidade de de frutas colhidas ($1 <= F <= 10^3$).

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

A entrada deverá ser lida de um arquivo texto chamado suco.in.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo um número real, escrito com precisão de duas casas decimais, representando o volume de suco, em litros, a que cada amigo tem direito.

A saída deverá ser escrita em um arquivo de texto chamado suco.out.

Exemplo de entrada	Saída para o exemplo de entrada
1 1 5 431 101 330 0 0	0.05 4.31 0.16

Problema D

Help!

Nome do arquivo fonte: help.java, help.php ou help.py

Bem, temos que admitir: precisamos da sua ajuda. Neste ano as coisas não estão correndo muito bem como gostaríamos, e não pudemos terminar o software do sistema de competição a tempo. Uma parte vital foi esquecida, e até onde sabemos, precisaremos do sistema funcionando esta tarde, para a competição oficial. A parte esquecida é o módulo que computa o placar dos times, dando a relação das suas submissões.

Por favor, nos ajude!

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um número inteiro N indicando o número de submissões neste teste (1 <= N <= 300). Cada uma das N linhas descreve uma submissão; cada uma dessas linhas contém um identificador de problema (uma letra de “A” a “Z”), seguida de um número inteiro T representando o tempo em minutos (0 <= T <= 300), seguido de um julgamento (a palavra “correct” (correto) ou a palavra “incorrect” (errado)). A entrada é ordenada em ordem ascendente de tempo, e haverá no máximo um julgamento “correct” (correto) para cada problema. O final do arquivo é indicado por N = 0.

A entrada deverá ser lida de um arquivo texto chamado help.in.

Saída

Para cada caso de teste de entrada seu programa deve imprimir uma única linha de saída contendo dois números inteiros S e P, separados por um espaço, onde o S é o número de problemas distintos com um julgamento “correct” (correto) e P é a somatória dos tempos que cada problema distinto foi julgado “correct” (correto) pela primeira vez, mais 20 para cada submissão “incorrect” (errada) de um problema que mais tarde tenha sido julgado “correct” (correto).

A saída deverá ser escrita em um arquivo texto chamado help.out.

Exemplo de entrada	Saída para o exemplo de entrada
3 A 120 incorrect A 130 incorrect A 200 incorrect 5 A 100 correct B 110 incorrect B 111 correct C 200 correct D 300 incorrect 0	0 0 3 431

BOA SORTE !