MARATONA DE PROGRAMAÇÃO

Informações Gerais

A) Sobre a prova

1) Cada equipe deve realizar o login no sistema de submissões de soluções (www.maratona.upf.br) com o login e senha passados. Após, a equipe deve proceder a troca de sua senha na opção “Options”
2) A prova é composta de seis (6) problemas que podem ser resolvidos em qualquer ordem;
3) Quando a equipe tiver a solução de um problema, deve acessar a opção “Runs”, escolhendo o problema, a linguagem e o código fonte da solução
4) O resultado da submissão da solução irá ser mostrado a equipe na opção “Runs”
5) Os seguintes resultados são possíveis em submissões
YES - Solução Aceita
NO – Compilation error
NO – Time limit exceed
NO – Presentation error
NO – Wrong Anwser
6) A cada problema julgado correto, a equipe irá receber um balão identificando o acerto do problema
7) Cada submissão errada terá a penalidade de 20m acrescido ao tempo da equipe. Somente os problemas considerados corretos terão a contagem da penalização
8) Quando faltar 1 hora para o encerramento da prova, o placar ficará congelado, porém balões serão distribuídos
9) Quando faltar 30 minutos para o final da prova, as equipes deixam de receber notificação de resultados de submissão

B) Sobre a entrada

1) A entrada de seu programa deve ser lida da entrada padrão.
2) A entrada é composta por vários casos de teste, cada um descrito em um número de linhas que depende do problema.
3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém o caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

C) Sobre a saída

1) A saída de seu programa deve ser escrita na saída padrão.
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter o caractere final-de-linha

Problema A

O Fim Está Próximo!

Balão Vermelho

Arquivo: fim.[c|cpp|java]

O conhecido mago Zak Galou fez um curso de final de semana sobre futurologia em uma instituição obscura. A partir dos conhecimentos adquiridos lá, Zak Galou inovou e passou a ministrar o seminário Como Aproveitar a Vida antes do Fim! ao redor do mundo. Casualmente, Zak Galou estará em Passo Fundo e região nesta semana. Como quem já o conhece sabe, Zak Galou é bom em matar monstros e nos trabalhos em vinhedos, mas é péssimo para resolver problemas.

Ele aproveitou sua passagem pela cidade e solicitou um programa que, dados um prazo limite para o final e uma série de atividades a realizar, cada qual com um tempo necessário de execução e um valor positivo de felicidade ao ser cumprida, calcula qual seria o valor máximo de felicidade que se poderia atingir.

Pelas suas qualidades como futurólogo, Zak Galou prevê o tempo limite para o final. Além disso, ele estabelece o tempo de execução e a felicidade gerada para cada atividade (plantar uma árvore, escrever um livro, tomar um sorvete de melancia etc.). Você se prontificou a ajudá-lo nesta tarefa de indicar o máximo de felicidade possível de se atingir ao cumprir algumas (eventualmente nenhuma ou todas) atividades sem estourar o tempo limite para o final.

Entrada

A entrada é composta por vários casos de teste. Para cada um deles, a primeira linha contém os inteiros L e N , separados por um espaço em branco, que indicam o tempo limite para o fim e o número de atividades indicadas por Zak Galou, respectivamente ($1 \leq L \leq 100$ e $1 \leq N \leq 100$). A próxima linha contém N inteiros Ti separados por um espaço em branco, que indicam o tempo de execução de cada tarefa ($1 \leq Ti \leq 100$, $1 \leq i \leq N$). A linha seguinte contém N inteiros Vi separados por um espaço em branco, que indicam a felicidade gerada ao concluir cada tarefa ($1 \leq Vi \leq 50$, $1 \leq i \leq N$). O final da entrada é indicado por L = N = 0.

Saída

Para cada caso de teste é escrita uma linha no formato: “Caso X gera felicidade Y ”, onde X indica o número do caso de teste (o primeiro lido na entrada será 1, o segundo será 2, e assim por diante) e Y indica a soma das felicidades Vi de modo que os tempos Ti não ultrapassem L.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
1 2 1 1 10 12 10 5 12 20 50 13 100 1 22 25 2 50 20 4 5 6 16 4 23 24 46 3 0 0	Caso 1 gera felicidade 12 Caso 2 gera felicidade 0 Caso 3 gera felicidade 50

Problema B
Guarda Costeira

Balão: Branco
Arquivo : Arquivo: `guarda.[c|cpp|java]`
`Timelimit: 1`

"Pega ladrão! Pega ladrão!" Roubaram a bolsa de uma inocente senhora que caminhava na praia da Nlogônia e o ladrão fugiu em direção ao mar. Seu plano parece obvio: ele pretende pegar um barco e escapar!

O fugitivo, que a essa altura já está a bordo de sua embarcação de fuga, pretende seguir perpendicularmente à costa em direção ao limite de águas internacionais, que fica a 12 milhas náuticas de distância, onde estará são e salvo das autoridades locais. Seu barco consegue percorrer essa distância a uma velocidade constante de VF nós.

A Guarda Costeira pretende interceptá-lo, e sua embarcação tem uma velocidade constante de VG nós. Supondo que ambas as embarcações partam da costa exatamente no mesmo instante, com uma distância de D milhas náuticas entre elas, será possível a Guarda Costeira alcançar o ladrão antes do limite de águas internacionais?

Assuma que a costa da Nlogônia é perfeitamente retilínea e o mar bastante calmo, de forma a permitir uma trajetória tão retilínea quanto a costa.

Entrada

A entrada é composta por diversos casos de teste e termina com final de arquivo (EOF). Cada caso de teste é descrito em um linha contendo três inteiros, **D** ($1 \leq$ **D** ≤ 100), **VF** ($1 \leq$ **VF** ≤ 100) e **VG** ($1 \leq$ **VG** ≤ 100), indicando respectivamente a distância inicial entre o fugitivo e a Guarda Costeira, a velocidade da embarcação do fugitivo e a velocidade da embarcação da Guarda Costeira.

Saída

Para cada caso de teste imprima uma linha contendo ‘S’ se for possível que a Guarda Costeira alcance o fugitivo antes que ele ultrapasse o limite de águas internacionais ou ‘N’ caso contrário.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
5 1 12 12 10 7 12 9 10 10 5 5 9 12 15	S N N N S

Problema C
Revisão de Contrato

Balão: Azul
Arquivo : Arquivo: `contrato.[c|cpp|java]`
`Timelimit: 1`

Durante anos, todos os contratos da Associação de Contratos da Modernolândia (ACM) foram datilografados em uma velha máquina de datilografia.

Recentemente Sr. Miranda, um dos contadores da ACM, percebeu que a máquina apresentava falha em um, e apenas um, dos dígitos numéricos. Mais especificamente, o dígito falho, quando datilografado, não é impresso na folha, como se a tecla correspondente não tivesse sido pressionada. Ele percebeu que isso poderia ter alterado os valores numéricos representados nos contratos e, preocupado com a contabilidade, quer saber, a partir dos valores originais negociados nos contratos, que ele mantinha em anotações manuscritas, quais os valores de fato representados nos contratos. Por exemplo, se a máquina apresenta falha no dígito 5, o valor 1500 seria datilografado no contrato como 100, pois o 5 não seria impresso. Note que o Sr. Miranda quer saber o valor numérico representado no contrato, ou seja, nessa mesma máquina, o número 5000 corresponde ao valor numérico 0, e não 000 (como ele de fato aparece impresso).

Entrada

A entrada consiste de diversos casos de teste, cada um em uma linha. Cada linha contém dois inteiros **D** e **N** ($1 \leq \mathbf{D} \leq 9$, $1 \leq \mathbf{N} < 10^{100}$), representando, respectivamente, o dígito que está apresentando problema na máquina e o número que foi negociado originalmente no contrato (que podem ser grande, pois Modernolândia tem sido acometida por hiperinflação nas últimas décadas).

O ultimo caso de teste é seguido por uma linha que contém apenas dois zeros separados por espaços em branco.

Saída

Para cada caso de teste da entrada o seu programa deve imprimir uma linha contendo um único inteiro **V**, o valor numérico representado de fato no contrato.

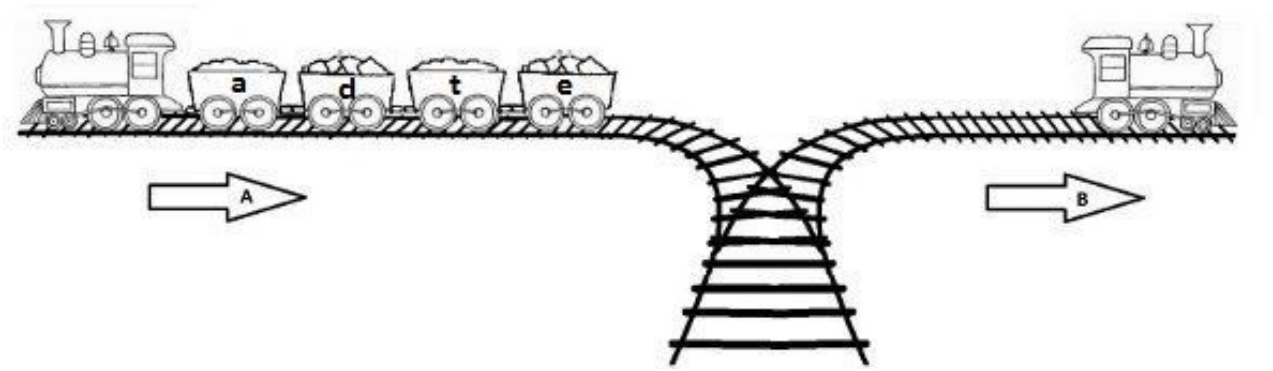
Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
5 5000000 3 123456 9 23454324543423 9 99999999991999999 7 777 0 0	0 12456 23454324543423 1 0

Problema D
Trilhos Novamente... Traçando Movimentos

Balão: Verde
Arquivo : Arquivo: trilhos.[c|cpp|java]
Timelimit: 1

Você lembra daquela estação de trem da cidade PopPush? Apenas para relembrar, existe uma estação de trem em um país incrivelmente acidentado. Além disso, a estação foi construída no século passado e infelizmente os fundos eram muito limitados. Em um determinado trecho foi possível construir apenas uma pista e, a solução encontrada para transportar as cargas nos dois sentidos foi construir uma estação que permitisse desconectar os vagões de uma locomotiva e conectar em outra, que iria em outro sentido.



Cada trem que chega na direção A é manobrado e seus vagões continuam na direção B, reorganizados conforme o chefe da estação deseja. Ao chegar pelo lado A, cada vagão é desconectado e vai até a estação e depois segue para a direção B, para ser conectado na segunda locomotiva. Você pode desconectar quantos trens deseja na estação, mas o vagão que entra na estação só pode sair pelo lado B e uma vez que ele sai, não pode mais entrar novamente.

Todos vagões são identificados pelas letras minúsculas (**a** até **z**). Isto significa 26 vagões no máximo. O chefe da organização dos vagões precisa agora que você ajude a resolver para ele, através de um programa, qual a sequência de movimentos é necessária para obter a saída desejada após a entrada na estação, seguindo para a direção B. O movimento de entrada e saída da estação é descrito respectivamente pelas letras **I** e **R** (Insere e Remove). Utilizando a figura dada como exemplo, a entrada e,t,d,a para uma saída desejada d,a,t,e, resulta nos movimentos I,I,I,R,I,R,R,R

Entrada

A entrada consiste em vários casos de teste, onde cada caso de teste é composto por 3 linhas. A primeira das 3 linhas contém um número inteiro **N** que representa o número total de vagões. A segunda linha contém a sequência dos vagões que vêm do lado A e a Terceira linha contém a sequência que o chefe de organização deseja como saída para o lado B. A última linha de entrada contém apenas 0, indicando o fim da entrada.

Saída

O arquivo de saída contém a quantidade de linhas correspondente ao número de casos de teste de entrada. Cada linha de saída contém uma sequência de **I** e **R** conforme o exemplo. Se não for possível mostrar a saída, as operações devem ser interrompidas e a mensagem "**Impossible**" deve ser impressa, com um espaço após a sequência.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 e t d a d a t e 5 o s t a p p a t o s 0	IIIRIRRR IIIIIRRR Impossible

Problema E
Zerinho ou Um

Balão: Amarelo
Arquivo : Arquivo: zerinho.[c|cpp|java]
Timelimit: 2

Todos devem conhecer o jogo Zerinho ou Um (em algumas regiões também conhecido como Dois ou Um), utilizado para determinar um ganhador entre três ou mais jogadores. Para quem não conhece, o jogo funciona da seguinte maneira. Cada jogador escolhe um valor entre zero ou um; a um comando (geralmente um dos competidores anuncia em voz alta “Zerinho ou... Um!”), todos os participantes mostram o valor escolhido, utilizando uma das mãos: se o valor escolhido foi um, o competidor mostra o dedo indicador estendido; se o valor escolhido foi zero, mostra a mão com todos os dedos fechados. O ganhador é aquele que tiver escolhido um valor diferente de todos os outros; se não há um jogador com valor diferente de todos os outros (por exemplo todos os jogadores escolhem zero, ou um grupo de jogadores escolhe zero e outro grupo escolhe um), não há ganhador. Alice, Beto e Clara são grandes amigos e jogam Zerinho a toda hora: para determinar quem vai comprar a pipoca durante a sessão de cinema, quem vai entrar na piscina primeiro, etc. Jogam tanto que resolveram fazer um plugin no Facebook para jogar Zerinho. Como não sabem programar, dividiram as tarefas entre amigos que sabem, inclusive você. Dados os três valores escolhidos por Alice, Beto e Clara, cada valor zero ou um, escreva um programa que determina se há um ganhador, e nesse caso determina quem é o ganhador.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste consiste de uma única linha, que contém três inteiros A, B e C (A,B,C só podem ser 0 ou 1), indicando respectivamente os valores escolhidos por Alice, Beto e Clara. O final da entrada é determinado por EOF (End of File).

Saída

Para cada caso de teste, seu programa deve produzir uma única linha, contendo um único caractere. Se o vencedor é Alice o caractere deve ser ‘A’, se o vencedor é Beto o caractere deve ser ‘B’, se o vencedor é Clara o caractere deve ser ‘C’ e se não há vencedor o caractere deve ser ‘*’ (asterisco).

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
1 1 0 0 0 0 1 0 0	C * A

Problema F
Ir e Vir

Balão: Azul escuro
Arquivo : Arquivo: irevir.[c|cpp|java]
Timelimit: 1

Numa certa cidade há **N** intersecções ligadas por ruas de mão única e ruas com mão dupla de direcão. É uma cidade moderna, de forma que muitas ruas atravessam túneis ou têm viadutos. Evidentemente é necessário que se possa viajar entre quaisquer duas intersecções, isto é, dadas duas intersecções **V** e **W**, deve ser possível viajar de **V** para **W** e de **W** para **V**.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros **N** e **M**, separados por um espaço em branco, indicando respectivamente o número de intersecções ($2 \leq \mathbf{N} \leq 2000$) e o número de ruas ($2 \leq \mathbf{M} \leq \mathbf{N}(\mathbf{N}-1)/2$). O caso de teste tem ainda mais **M** linhas, que contêm, cada uma, uma descrição de cada uma das **M** ruas. A descrição consiste de três inteiros **V**, **W** e **P**, separados por um espaço em branco, onde **V** e **W** são identificadores distintos de intersecções ($1 \leq \mathbf{V}, \mathbf{W} \leq \mathbf{N}$, $\mathbf{V} \neq \mathbf{W}$) e **P** pode ser 1 ou 2; se **P** = 1 então a rua é de mão única, e vai de **V** para **W**; se **P** = 2 então a rua é de mão dupla, liga **V** e **W**. Não existe duas ruas ligando as mesmas intersecções.

Saída

Para cada caso de teste seu programa deve imprimir uma linha contendo um inteiro G, onde G é igual a 1 se o requisito de conexidade está satisfeito, ou G é igual a 0, caso contrário.

Exemplos

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