acm International Collegiate Programming Contest 2010 IBM event sponsor

Maratona de Programação da SBC 2010

Sub-Regional Brasil do ACM ICPC

18 de Setembro de 2010

Caderno de Problemas

Instruções

1) Este caderno contém 11 problemas; as páginas estão numeradas de 1 a 18, não contando esta página de rosto. Verifique se o caderno está completo.
2) Em todos os problemas, a entrada de seu programa deve ser lida da *entrada padrão*. A saída deve ser escrita na *saída padrão*.

Promoção:

SBC Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

v1.41

Problema A
Revisão de Contrato

Nome do arquivo fonte: `contrato.c`, `contrato.cpp` *ou* `contrato.java`

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 ($1 \leq D \leq 9$, $1 \leq N < 10^{100}$), 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 último 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.

Exemplo de entrada	**Exemplo de saída**
5 5000000	0
3 123456	12456
9 23454324543423	23454324543423
9 99999999991999999	1
7 777	0
0 0	

Problema B
Robô Colecionador

Nome do arquivo fonte: `robo.c`, `robo.cpp` *ou* `robo.java`

Um dos esportes favoritos na Robolândia é o Rali dos Robôs. Este rali é praticado em uma arena retangular gigante de N linhas por M colunas de células quadradas. Algumas das células estão vazias, algumas contêm figurinhas da Copa (muito apreciadas pelas inteligências artificiais da Robolândia) e algumas são ocupadas por pilastras que sustentam o teto da arena. Em seu percurso os robôs podem ocupar qualquer célula da arena, exceto as que contém pilastras, que bloqueiam o seu movimento. O percurso do robô na arena durante o rali é determinado por uma sequência de instruções. Cada instrução é representada por um dos seguintes caracteres: '`D`', '`E`' e '`F`', significando, respectivamente, "gire 90 graus para a direita", "gire 90 graus para a esquerda" e "ande uma célula para a frente". O robô começa o rali em uma posição inicial na arena e segue fielmente a sequência de instruções dada (afinal, eles são robôs!). Sempre que o robô ocupa uma célula que contém uma figurinha da Copa ele a coleta. As figurinhas da Copa não são repostas, ou seja, cada figurinha pode ser coletada uma única vez. Quando um robô tenta andar para uma célula onde existe uma pilastra ele patina, permanecendo na célula onde estava, com a mesma orientação. O mesmo também acontece quando um robô tenta sair da arena.

Dados o mapa da arena, descrevendo a posição de pilastras e figurinhas, e a sequência de instruções de um robô, você deve escrever um programa para determinar o número de figurinhas coletadas pelo robô.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três números inteiros N, M e S ($1 \leq N, M \leq 100$, $1 \leq S \leq 5 \times 10^4$), separados por espaços em branco, indicando respectivamente o número de linhas e o número de colunas da arena e o número de instruções para o robô. Cada uma das N linhas seguintes da entrada descreve uma linha de células da arena e contém uma cadeia com M caracteres. A primeira linha que aparece na descrição da arena é a que está mais ao Norte; a primeira coluna que aparece na descrição de uma linha de células da arena é a que está mais a Oeste.

Cada célula da arena pode conter um dos seguintes caracteres:

- '`.`' — célula normal;
- '`*`' — célula que contém uma figurinha da Copa;
- '`#`' — célula que contém uma pilastra;
- '`N`', '`S`', '`L`', '`O`' — célula onde o robô inicia o percurso (única na arena). A letra representa a orientação inicial do robô (Norte, Sul, Leste e Oeste, respectivamente).

A última linha da entrada contém uma sequência de S caracteres dentre '`D`', '`E`' e '`F`', representando as instruções do robô.

O último caso de teste é seguido por uma linha que contém apenas três números zero separados por um espaço em branco.

Saída

Para cada rali descrito na entrada seu programa deve imprimir uma única linha contendo um único inteiro, indicando o número de figurinhas que o robô colecionou durante o rali.

Exemplo de entrada	**Exemplo de saída**
3 3 2 *** *N* *** DE 4 4 5 ...# *#O. *.*. *.#. FFEFF 10 10 20 ....*..... ....*.. .....*.... ..*.#..... ...#N.*..* ...*...... .... FDFFFFFFEEFFFFFFEFDF 0 0 0	0 1 3

Problema C
Livro-caixa

Nome do arquivo fonte: `caixa.c`, `caixa.cpp` *ou* `caixa.java`

A FCC (Fundação de Combate à Corrupção) desmontou um grande esquema de corrupção na Nlogônia. Durante a operação, foram apreendidos diversos cadernos e livros com anotações documentando as transações ilícitas realizadas pelo esquema.

Vários desses livros contém páginas com os valores de várias transações em nilogos (a moeda local da Nlogônia, cujo símbolo é N$) e o fluxo de caixa resultante dessas transações. Por exemplo, se em uma página foi registrada uma entrada de N$ 7, uma entrada de N$ 2, uma saída de N$ 3, uma entrada de N$ 1 e outra saída de N$ 11, o fluxo de caixa nesta página é $7 + 2 - 3 + 1 - 11 = -4$.

No entanto, para dificultar o trabalho da polícia, os contraventores não anotaram em seus livros qual o tipo de cada transação. No exemplo acima, as anotações na página seriam apenas 7, 2, 3, 1 e 11 (sem indicação se elas são entradas ou saídas). O fluxo de caixa de cada página sempre é anotado normalmente, com o sinal (no caso, -4).

Para obter a condenação dos contraventores, os promotores precisam poder afirmar com certeza se cada operação foi uma entrada ou uma saída. No exemplo acima, a transação de N$ 7 certamente foi uma entrada, e a transação de N$ 11 certamente foi uma saída. Mas, não se pode afirmar nada sobre as transações de N$ 2, N$ 3, e N$ 1. As transações de N$ 2 e N$ 1 poderiam ter sido entradas e a transação de N$ 3 uma saída, ou N$ 2 e N$ 1 poderiam ter sido saídas e a transação de N$ 3 uma uma entrada.

Muitos cadernos possuem números relativamente grandes, com muitas transações, então é difícil para a polícia reconstruir o histório de operações. Por isso, eles precisam de um programa que o faça de forma eficiente.

Entrada

A entrada consiste de vários casos de teste. A primeira linha da entrada contém dois inteiros N e F, indicando respectivamente o número de operações na página ($2 \leq N \leq 40$) e o fluxo de caixa para esta página ($-16000 \leq F \leq 16000$). Cada uma das N linhas seguintes contém um inteiro T_i indicando o valor da i-ésima transação ($1 \leq T_i \leq 1000$).

O último 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 seu programa deve imprimir uma única linha com N caracteres. O i-ésimo caractere deve ser '`+`', se for possível afirmar com certeza que a i-ésima operação foi uma entrada, '`-`', se for possível afirmar com certeza que a i-ésima operação foi uma saída, e '`?`', se não for possível determinar com certeza qual o tipo da operação.

Caso não exista uma sequência de entradas e saídas que totalize o fluxo de caixa indicado, imprima uma única linha contendo o caractere '`*`'.

Exemplo de entrada	**Exemplo de saída**
5 7 1 2 3 4 5 4 15 3 5 7 11 5 12 6 7 7 1 7 0 0	?+??+ * +??-?

Problema D
Desvio de Rota

Nome do arquivo fonte: `desvio.c`, `desvio.cpp` *ou* `desvio.java`

O sistema rodoviário de um país interliga todas as suas N cidades de modo que, a partir de uma cidade qualquer, é possível chegar a cada uma das outras cidades trafegando pelas estradas existentes. Cada estrada liga duas cidades distintas, tem mão dupla e um único posto de pedágio (o pedágio é pago nos dois sentidos de tráfego). As estradas não se intersectam a não ser nas cidades. Nenhum par de cidades é interligado por duas ou mais estradas.

A Transportadora Dias oferece um serviço de transporte de encomendas entre as cidades. Cada encomenda deve ser levada de uma cidade A para uma outra cidade B. A direção da Transportadora Dias define, para cada encomenda, uma *rota de serviço*, composta por C cidades e $C-1$ estradas: a primeira cidade da rota de serviço é a origem da encomenda, a última o destino da encomenda. A rota de serviço não passa duas vezes pela mesma cidade, e o veículo escolhido para fazer o transporte de uma encomenda pode trafegar apenas pela rota de serviço definida.

Certo dia, no entanto, o veículo que executava uma entrega quebrou e precisou ser levado para conserto em uma cidade que não está entre as cidades de sua rota de serviço. A direção da Transportadora Dias quer saber qual é o menor custo total, em termos de pedágio, para que o veículo entregue a encomenda na cidade destino, a partir da cidade em que foi consertado, mas com uma restrição adicional: se em algum momento o veículo passar por uma das cidades que compõem a sua rota de serviço, ele deve voltar a obedecer a rota de serviço.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros N, M, C e K ($4 \leq N \leq 250, 3 \leq M \leq N \times (N-1)/2$, $2 \leq C \leq N-1$ e $C \leq K \leq N-1$), representando, respectivamente, o número de cidades do país, o número de estradas, o número de cidades na rota de serviço e a cidade em que o veículo foi consertado. As cidades são identificadas por inteiros de 0 a $N-1$. A rota de serviço é $0, 1, \ldots, C-1$, ou seja, a origem é 0, de 0 passa para 1, de 1 para 2 e assim por diante, até o destino $C-1$.

As M linhas seguintes descrevem o sistema rodoviário do país. Cada uma dessas linhas descreve uma estrada e contém três inteiros U, V e P ($0 \leq U, V \leq N-1$, $U \neq V$, $0 \leq P \leq 250$), indicando que há uma estrada interligando as cidades U e V com custo de pedágio P. O último caso de teste é seguido por uma linha contendo quatro zeros separados por espaço em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha, contendo um único inteiro T, o custo total mínimo necessário, em termos de pedágio, para que o veículo chegue ao destino.

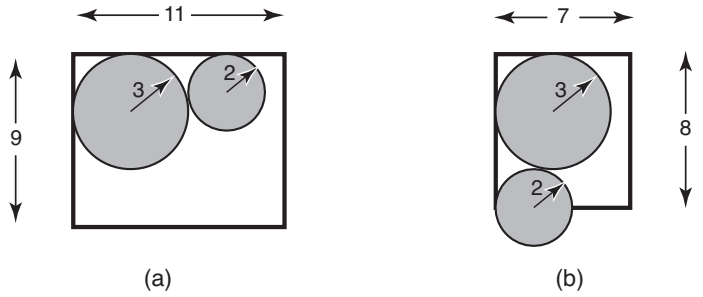
Exemplo de entrada	**Exemplo de saída**
4 6 3 3 0 1 10 1 2 10 0 2 1 3 0 1 3 1 10 3 2 10 6 7 2 5 5 2 1 2 1 10 1 0 1 3 0 2 3 4 2 3 5 3 5 4 2 5 5 2 4 0 1 1 1 2 2 2 3 3 3 4 4 4 0 5 0 0 0 0	10 6 6

Problema E
Elevador

Nome do arquivo fonte: `elevador.c`, `elevador.cpp` *ou* `elevador.java`

A FCC (Fábrica de Cilindros de Carbono) fabrica de vários tipos de cilindros de carbono. A FCC está instalada no décimo andar de um prédio, e utiliza os vários elevadores do prédio para transportar os cilindros. Por questão de segurança, os cilindros devem ser transportados na posição vertical; como são pesados, no máximo dois cilindros podem ser transportados em uma única viagem de elevador. Os elevadores têm formato de paralelepípedo e sempre têm altura maior que a altura dos cilindros.

Para minimizar o número de viagens de elevador para transportar os cilindros, a FCC quer, sempre que possível, colocar dois cilindros no elevador. A figura abaixo ilustra, esquematicamente (vista superior), um caso em que é isto possível (a), e um caso em que isto não é possível (b):



Como existe uma quantidade muito grande de elevadores e de tipos de cilindros, a FCC quer que você escreva um programa que, dadas as dimensões do elevador e dos dois cilindros, determine se é possível colocar os dois cilindros no elevador.

Entrada

A entrada contém vários casos de teste. A primeira e única linha de cada caso de teste contém quatro números inteiros L, C, R_1 e R_2, separados por espaços em branco, indicando respectivamente a largura do elevador ($1 \leq L \leq 100$), o comprimento do elevador ($1 \leq C \leq 100$), e os raios dos cilindros ($1 \leq R_1, R_2 \leq 100$).

O último caso de teste é seguido por uma linha que contém quatro zeros separados por espaços em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha com um único caractere: '`S`' se for possível colocar os dois cilindros no elevador e '`N`' caso contrário.

Exemplo de entrada	**Exemplo de saída**

`11 9 2 3`
`7 8 3 2`
`10 15 3 7`
`8 9 3 2`
`0 0 0 0`
	`S`
	`N`
	`N`
	`S`

Problema F
Fórmula 1

Nome do arquivo fonte: `formula.c`, `formula.cpp` *ou* `formula.java`

A temporada de Fórmula 1 consiste de uma série de corridas, conhecidas como Grandes Prêmios, organizados pela Federação Internacional de Automobilismo (FIA). Os resultados de cada Grande Prêmio são combinados para determinar o Campeonato Mundial de Pilotos. Mais especificamente, a cada Grande Prêmio são distribuídos pontos para os pilotos, dependendo da classificação na corrida. Ao final da temporada, o piloto que tiver somado o maior número de pontos é declarado Campeão Mundial de Pilotos.

Os organizadores da Fórmula 1 mudam constantemente as regras da competição, com o objetivo de dar mais emoção às disputas. Uma regra modificada para a temporada de 2010 foi justamente a distribuição de pontos em cada Grande Prêmio. Desde 2003 a regra de pontuação premiava os oito primeiros colocados, obedecendo a seguinte tabela:

Colocação	1	2	3	4	5	6	7	8
Pontos	10	8	6	5	4	3	2	1

Ou seja, o piloto vencedor ganhava 10 pontos, o segundo colocado ganhava 8 pontos, e assim por diante.

Na temporada de 2010 os dez primeiros colocados receberão pontos, obedecendo a seguinte tabela:

Colocação	1	2	3	4	5	6	7	8	9	10
Pontos	25	18	15	12	10	8	6	4	2	1

A mudança no sistema de pontuação provocou muita especulação sobre qual teria sido o efeito nos Campeonatos Mundiais passados se a nova pontuação tivesse sido utilizada nas temporadas anteriores. Por exemplo, teria Lewis Hamilton sido campeão em 2008, já que a diferença de sua pontuação total para Felipe Massa foi de apenas um ponto? Para acabar com as especulações, a FIA contratou você para escrever um programa que, dados os resultados de cada corrida de uma temporada determine Campeão Mundial de Pilotos para sistemas de pontuações diferentes.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros G e P separados por um espaço em branco, indicando respectivamente o número de Grandes Prêmios ($1 \leq G \leq 100$) e o número de pilotos ($1 \leq P \leq 100$). Os pilotos são identificados por inteiros de 1 a P. Cada uma das G linhas seguintes indica o resultado de uma corrida, e contém P inteiros separados por espaços em branco. Em cada linha, o i-ésimo número indica a ordem de chegada do pilodo i na corrida (o primeiro número indica a ordem de chegada do piloto 1 naquela corrida, o segundo número indica a ordem de chegada do piloto 2 na corrida, e assim por diante). A linha seguinte contém um único número inteiro S indicando o número de sistemas de pontuação ($1 \leq S \leq 10$), e após, cada uma das S linhas seguintes contém a descrição de um sistema de pontuação. A descrição de um sistema de pontuação inicia com um inteiro K ($1 \leq K \leq P$), indicando a última ordem de chegada que receberá

pontos, seguido de um espaço em branco, seguido de K inteiros $k_0, k_1, \ldots, k_{n-1}$ ($1 \leq k_i \leq 100$) separados por espaços em branco, indicando os pontos a serem atribuídos (o primeiro inteiro indica os pontos do primeiro colocado, o segundo inteiro indica os pontos do segundo colocado, e assim por diante).

O último caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

Para cada caso de sistema de pontuação da entrada seu programa deve imprimir uma linha, que deve conter o identificador do Campeão Mundial de Pilotos. Se houver mais de um Campeão Mundial Pilotos (ou seja, se houver empate), a linha deve conter todos os Campeões Mundiais de Pilotos, em ordem crescente de identificador, separados por um espaço em branco.

Exemplo de entrada	**Exemplo de saída**
1 3 3 2 1 3 3 5 3 2 3 5 3 1 3 1 1 1 3 10 1 2 3 4 5 6 7 8 9 10 10 1 2 3 4 5 6 7 8 9 9 10 1 2 3 4 5 6 7 8 2 5 5 4 3 2 1 3 10 5 1 2 4 1 3 4 2 4 1 3 2 2 3 3 2 1 3 5 4 2 0 0	3 3 1 2 3 3 3 2 4 4

Problema G
Guerra nas estrelas

Nome do arquivo fonte: `guerra.c`, `guerra.cpp` *ou* `guerra.java`

Há muito tempo atrás, em uma galáxia muito, muito distante, havia um império que dominava a todos. Uma aliança rebelde, descontente com essa situação, decidiu lutar contra tais forças, com o objetivo de restaurar a democracia e a paz para todos os povos.

Capitão Cael, um dos comandantes rebeldes, está navegando pelo espaço com seu cruzador espacial, quando de repente percebe a presença de uma nave do Império (de acordo com os padrões estéticos da época, todas as naves são tetraedros). Após ser surpreendido por um ataque inicial do império, Cael percebe que está em posição de disparo e que pode posicionar um canhão em qualquer ponto de sua nave.

Como a potência de sua arma é fixa, Cael quer posicionar seu canhão de forma que a distância percorrida pelo feixe de energia até a nave do Império seja mínima, para evitar perdas. Para isso, ele pediu para que você, sub-capitão Cin Talig, calculasse a menor distância entre a nave rebelde e a nave do Império.

Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro T, indicando o número de casos de teste da entrada. Cada um dos T casos de teste é composto de oito linhas, cada uma descrevendo a coordenada de um vértice de uma nave; as quatro primeiras linhas indicam os vértices da nave rebelde; as quatro linhas seguintes indicam os vértices da nave do Império.

Cada descrição de coordenada é uma linha contendo três inteiros X, Y, Z indicando a coordenada do vértice no espaço ($-10^3 \leq X \leq 10^3, -10^3 \leq Y \leq 10^3, -10^3 \leq Z \leq 10^3$); os quatro vértices de cada nave sempre definem um tetraedro de volume não nulo e as duas naves são sempre disjuntas.

Saída

Para cada caso de teste da entrada seu programa deve imprimir um único número, indicando a distância entre as duas naves, com duas casas decimais de precisão. A distância entre as duas naves é sempre maior que zero.

Exemplo de entrada	**Exemplo de saída**
3 2 -1 -1 0 -1 -3 1 1 -4 1 1 -2 0 5 -1 2 5 1 1 3 2 1 3 0 1 0 -6 -5 0 -4 -2 6 -5 -2 2 -2 1 0 3 -5 0 5 -2 2 7 -2 -4 7 4 -4 -2 -2 -4 -4 1 4 -3 1 0 -4 -2 4 -1 4 4 1 1 -4 0 1 0 -1	2.83 6.03 1.90

Problema H
Plágio Musical

Nome do arquivo fonte: `plagio.c`, `plagio.cpp` *ou* `plagio.java`

As notas musicais são unidades básicas da música ocidental tradicional. Cada nota está associada a uma frequência. Duas notas musicais cujas frequências fundamentais tenham uma relação de potência de 2 (uma metade da outra, uma duas vezes a outra, etc.) são percebidas como muito similar. Por isso, todas as notas com esse tipo de relação recebem o mesmo nome, como descrito a seguir.

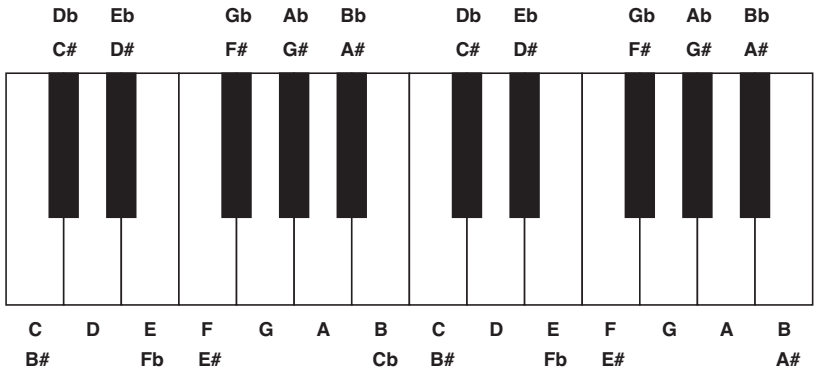
Há doze notas básicas, em uma sequência crescente de frequências, cada nota separada da anterior por uma mesma distância na escala musical (essa distância é chamada de *meio-tom*). Sete dessas doze notas são representadas por letras do alfabeto (A, B, C, D, E, F e G). A tabela abaixo mostra a distância, em meio-tons, entre essas notas.

Notas	A-B	B-C	C-D	D-E	E-F	F-G	G-A
Número de meios-tons	2	1	2	2	1	2	2

Note que há cinco notas que não são representadas pelas letras do alfabeto: as que estão entre A e B, entre C e D, entre D e E, entre F e G e entre G e A.

As notas podem ser modificadas por duas *alterações cromáticas*: *sustenido* e *bemol*, representadas respectivamente pelos símbolos '`#`' e '`b`'. Sustenido altera a nota em meio tom para cima, e bemol altera a nota em meio tom para baixo. Uma nota com alteração cromática é denotada pelo nome da nota seguida pelo símbolo da alteração. Note que com esse esquema conseguimos representar todas as doze notas.

A figura abaixo ilustra o nome das notas, segundo o esquema descrito acima, em um trecho de teclado de piano.



Uma melodia pode ser representada por uma sequência de *notas musicais*. Por exemplo,

```
A A D C# C# D E E E F# A D G# A
```

é uma melodia muito conhecida. Note no entanto que, como as distâncias entre os meios-tons são sempre iguais, a mesma melodia pode ser escrita iniciando em outra nota (dizemos que a melodia está em outro tom):

```
B B E D# D# E Gb Gb Gb G# B E A# B
```

Sua vizinha é uma famosa compositora que suspeita que tenham plagiado uma de suas músicas. Ela pediu a sua ajuda para escrever um programa que, dada a sequência de notas da melodia de sua música, e a sequência de notas de um trecho de melodia suspeito, verifique se o trecho supeito ocorre, em algum tom, na música dada.

Entrada

A entrada é composta por vários casos de teste. A primeira linha de um caso de teste contém dois inteiros M e T ($1 \leq M \leq 100000$, $1 \leq T \leq 10000$, $T \leq M$), indicando respectivamente o número de notas da música e do trecho suspeito de ter sido plagiado. As duas linhas seguintes contém M e T notas, respectivamente, indicando as notas da música e do trecho suspeito.

As notas em cada linha são separadas por espaço; cada nota é uma dentre '`A`', '`B`', '`C`', '`D`', '`E`', '`F`' ou '`G`', possivelmente seguida de um modificador: '`#`' para um sustenido ou '`b`' para um bemol.

O último caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

Para cada caso de teste, imprima uma única linha contendo um caractere: '`S`' caso o trecho realmente tenha sido plagiado pela música ou '`N`' caso contrário.

Exemplo de entrada	**Exemplo de saída**
16 4 D G A B C D G G G C D E F# G C C G G C D 12 2 C C# D D# E F F# G G# A A# B C D 12 2 C Db D Eb E F Gb G Ab A Bb B C D 4 3 C E G Bb D F# A 0 0	S N N S

Problema I
Ir e Vir

Nome do arquivo fonte: `ir.c`, `ir.cpp` *ou* `ir.java`

Numa certa cidade há N intersecções ligadas por ruas de mão única e ruas com mão dupla de direção. 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 posssível viajar de V para W e de W para V.

Sua tarefa é escrever um programa que leia a descrição do sistema de tráfego de uma cidade e determine se o requisito de conexidade é satisfeito ou não.

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 interseções ($2 \leq N \leq 2000$) e o número de ruas ($2 \leq M \leq N(N-1)/2$). 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 ($1 \leq V, W \leq N$, $V \neq W$) 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.

O último caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

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

Exemplo de entrada	**Exemplo de saída**

`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`

Problema J
Leitura Ótica

Nome do arquivo fonte: `leitura.c`, `leitura.cpp` *ou* `leitura.java`

O professor João decidiu aplicar somente provas de múltipla escolha, para facilitar a correção. Em cada prova, cada questão terá cinco alternativas (A, B, C, D e E), e o professor vai distribuir uma folha de resposta para cada aluno. Ao final da prova, as folhas de resposta serão escaneadas e processadas digitalmente para se obter a nota de cada aluno. Inicialmente, ele pediu ajuda a um sobrinho, que sabe programar muito bem, para escrever um programa para extrair as alternativas marcadas pelos alunos nas folhas de resposta. O sobrinho escreveu uma boa parte do software, mas não pode terminá-lo, pois precisava treinar para a Maratona de Programação.

Durante o processamento, a prova é escaneada usando tons de cinza entre 0 (preto total) e 255 (branco total). Após detectar os cinco retângulos correspondentes a cada uma das alternativas, ele calcula a média dos tons de cinza de cada pixel, retornando um valor inteiro correspondente àquela alternativa. Se o quadrado foi preenchido corretamente o valor da média é zero (preto total). Se o quadrado foi deixado em branco o valor da média é 255 (branco total). Assim, idealmente, se os valores de cada quadrado de uma questão são (255, 0, 255, 255, 255), sabemos que o aluno marcou a alternativa B para essa questão. No entanto, como as folhas são processadas individualmente, o valor médio de nível de cinza para o quadrado totalmente preenchido não é necessariamente 0 (pode ser maior); da mesma foram, o valor para o quadrado não preenchido não é necessariamente 255 (pode ser menor). O prof. João determinou que os quadrados seriam divididos em duas classes: aqueles com média menor ou igual a 127 serão considerados pretos e aqueles com média maior a 127 serão considerados brancos.

Obviamente, nem todas as questões das folhas de resposta são marcadas de maneira correta. Pode acontecer de um aluno se enganar e marcar mais de uma alternativa na mesma questão, ou não marcar nenhuma alternativa. Nesses casos, a resposta deve ser desconsiderada.

O professor João necessita agora de um voluntário para escrever um programa que, dados os valores dos cinco retângulos correspondentes às alternativas de uma questão determine qual a alternativa corretamente marcada, ou se a resposta à questão deve ser desconsiderada.

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 questões da folha de respostas ($1 \leq N \leq 255$). Cada uma das N linhas seguintes descreve a resposta a uma questão e contém cinco números inteiros A, B, C, D e E, indicando os valores de nível de cinza médio para cada uma das alternativas da resposta ($0 \leq A, B, C, D, E \leq 255$).

O último caso de teste é seguido por uma linha que contém apenas um número zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir N linhas, cada linha correspondendo a uma questão. Se a resposta à questão foi corretamente preenchida na folha de resposta, a linha deve conter a alternativa marcada (‘`A`’, ‘`B`’, ‘`C`’, ‘`D`’ ou ‘`E`’). Caso contrário, a linha deve conter o caractere ‘`*`’ (asterisco).

Exemplo de entrada	**Exemplo de saída**
3 0 255 255 255 255 255 255 255 255 0 255 255 127 255 255 4 200 200 200 0 200 200 1 200 200 1 1 2 3 4 5 255 5 200 130 205 0	A E C D * * B

Problema K
Jogo da Velha

Nome do arquivo fonte: `velha.c`, `velha.cpp` *ou* `velha.java`

O *jogo da velha* é um dos jogos mais antigos da humanidade; os primeiros registros dele são do século I antes de Cristo, no Império Romano. João e Maria jogam bastante jogo da velha, mas depois de algum tempo eles decidiram jogar uma variante do jogo da velha tradicional, o jogo da velha 1-D.

O jogo da velha 1-D é um jogo disputado por dois jogadores em um tabuleiro $1 \times N$; inicialmente, todas as casas do tabuleiro estão vazias. Os jogadores alternam-se desenhando uma cruz sobre uma casa vazia. O primeiro jogador a completar uma sequência de três ou mais cruzes em casas consecutivas ganha o jogo.

Maria logo percebeu que, dependendo da situação do jogo, sendo sua vez de jogar, ela pode sempre garantir a vitória, independente das jogadas de João. Isto é relativamente fácil para tabuleiros menores, mas para tabuleiros maiores, mesmo após várias jogadas, esta tarefa é mais difícil; por isso, ela pediu que você escrevesse um programa que, dada a situação do tabuleiro, decide se ela tem uma estratégia vencedora.

Entrada

A entrada contém vários casos de teste. A primeira linha de caso de teste contém um inteiro N, indicando o tamanho do tabuleiro ($3 \leq N \leq 10^4$). A linha seguinte contém uma sequência de N caracteres indicando quais casas do tabuleiro já foram ocupadas: um '`.`' indica que a casa correspondente está vazia, enquanto um '`X`' indica que a casa já teve uma cruz desenhada sobre ela. A entrada nunca contém três '`X`' consecutivos.

O último caso de teste é seguido por uma linha que contém um único número zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha contendo um único caractere: '`S`' caso Maria possua uma estratégia vencedora e '`N`' caso contrário.

Exemplo de entrada	**Exemplo de saída**

`5`
`.....`
`5`
`..X..`
`6`
`X.X.X.`
`12`
`............`
`0`
	`S`
	`N`
	`S`
	`N`