Jogos Olímpicos

2^a Seletiva Interna – 2016/2

Servidor BOCA:

`http://10.20.107.207/boca/` (acesso interno)

`http://200.19.107.207/boca/` (acesso externo)

Organização e Realização:

Claudio Cesar de Sá (coordenação geral), Gabriel Hermann Negri (coordenação técnica), Diego Buchinger (coordenação técnica), Rogério Eduardo da Silva (coordenação técnica), Lucas Hermann Negri (revisão).

Patrocinador 2016: LINX

Lembretes:

- Aos *javaneiros*: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido**. Ex: **classe petrus**, nome do arquivo **petrus.java**;
- Exemplo de leitura de entradas que funcionam:

```
Java: (import java.util.Scanner)
Scanner in = new Scanner(System.in);
ou
Scanner stdin = new Scanner(new BufferedReader(new InputStreamReader(System.in)));

C: (#include <stdio.h>)
int integer1; scanf("%d", &integer1);

C++: (#include <iostream>)
int integer1; std::cin >> integer1;
```

  Exemplo de saída de entradas:

```
Java: System.out.format("%d %d\n", integer1, integer2);
C: printf("%d %d\n", integer1, integer2);
C++: std::cout << integer1 << " " << integer2 << std::endl;
```

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, **siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos**. Deve-se considerar entradas e saídas padrão;
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o *clarification* para dúvidas da prova. Os juízes podem **opcionalmente** atendê-lo com respostas acessíveis a todos;
- As interfaces gráfica nas máquinas Linux são: Pantheon, OpenBox, LXDE (recomendável) e Unity. Usuário: *udesc*, sem e senha;
- Ao pessoal local: **cuidado com os pés sobre as mesas para não desligarem nenhum estabilizador/computador de outras equipes**!

Patrocinador e Agradecimentos

- LINX – Patrocinador oficial do ano de 2016;
- DCC/UDESC;
- Aos bolsistas deste ano pelo empenho;
- Alguns, muitos outros anônimos.

Jogos Olímpicos

2^a Seletiva Interna da UDESC 2016

20 de agosto de 2016

Conteúdo

Atenção quanto aos nomes e números dos problemas!!!

1 Problema A: Ameaça às Olimpíadas

`Arquivo: ameaca.[c|cpp|java]` `Tempo limite: 2 s`

Fonte:
www.clipartpanda.com

Jaime Bonde, o espião brasileiro, tem um trabalho muito perigoso: se infiltrar na Vila Olímpica em busca de evidências que possam levar a atentados terroristas. Ao entrar na Vila Olímpica, Jaime procura por evidências em todos prédios, movimentando-se pelos corredores e ruas menos vigiados para passar despercebido.

Para maximizar sua chance de encontrar evidências sem ser detectado pelos terroristas, Jaime pediu sua colaboração: ele passará para você a estrutura da Vila Olímpica e vai perguntar (uma ou mais perguntas por setor) qual é a chance de ser detectado por um terrorista ao passar de um prédio para outro, assumindo que voce achou o trajeto ótimo.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é iniciado por dois inteiros N ($0 < N \leq 200$) e M ($0 < M \leq 10000$), sendo que $N = 0$ e $M = 0$ marcam o final da entrada (N é o número de prédios da Vila Olímpica, e M é o número de corredores ou ruas). Cada uma das M linhas a seguir descrevem um corredor ou rua. A descrição de um corredor é composta por 3 inteiros, `A B C`, onde C ($0 \leq C \leq 100$) é a porcentagem de chance de Jaime ser detectado ao passar pelo corredor ou rua que liga os prédios A e B (sentido duplo) ($1 \leq A$, $B \leq N$). Todos prédios podem ser alcançados por meio dos corredores ou ruas, e não há mais de um corredor ou rua entre um par de prédios. Após a descrição dos corredores ou ruas existe um inteiro Q ($1 < Q \leq N$) que representa o número de perguntas. Cada pergunta é representada em uma linha contendo dois inteiros D P, marcando prédio de origem e o prédio de destino ($1 \leq D$, $P \leq N$). Após estas Q *perguntas* é que cada caso de teste se encerra.

Saída

Para cada pergunta em cada caso de teste, imprima a porcentagem (arredondada para duas casas decimais), do espião ser detectado no trajeto entre o par de prédios em questão (siga a formatação vista no exemplo abaixo). Deixe uma linha em branco após cada caso de teste.

Exemplo de entrada	**Exemplo de saída**
3 3 1 2 2 2 3 3 1 3 5 1 1 3 5 8 1 2 3 1 3 10 1 4 5 2 3 2 4 3 15 2 5 7 3 5 5 4 5 5 2 1 5 2 4 0 0	4.94% 9.69% 7.85%

2 Problema B: Brownies, Biscoitos e Refri

`Arquivo: brownie.[c|cpp|java]` `Tempo limite: 2 s`

Todo semestre o professor Claudius promove uma competição de programação na universidade onde leciona as quais são chamadas de "Seletivas". Esta competição é aberta para todos que tiverem interesse em participar. Obviamente, todos que participam desta competição querem testar e aprimorar as suas habilidades de programação e de resolução de problemas. Contudo, outro benefício que os participantes estão interessados é o *coffe-break* que o professor Claudius organiza durante a competição.

Fonte: www.eatfunfoods.com

O *coffe-break* geralmente está repleto de biscoitos e refrigerante para saciar a fome e a sede dos participantes mas, são os brownies do professor Claudius que realmente deixam todos agitados. Os brownies são sempre os primeiros a desaparecerem da mesa de comidas e nem sempre sobra um pedaço para cada participante, mesmo que todos sejam educados e peguem apenas um pedaço de brownie. Desta vez, contudo, professor Claudius teve uma ideia mirabolante: quando um grupo de participantes se aproxima da mesa de *coffe-break*, professor Claudius conta quantos brownies restam, e caso o número de brownies seja menor ou igual ao número de participantes que estão se aproximando, ele corta cada brownie pela metade. Se por acaso o número de brownies ainda for menor ou igual ao número de participantes, professor Claudius corta cada brownie pela metade novamente. Caso existam 3 pedaços de brownies restantes e um grupo de 24 participantes se aproxima da mesa de *coffe-break*, por exemplo, professor Claudius cortará os pedaços de brownies quatro vezes consecutivas ($3 \rightarrow 6 \rightarrow 12 \rightarrow 24 \rightarrow 48$) de modo que haverá brownies para todos e ainda sobrará ao menos um pedaço de brownie.

Professor Claudius gostaria de manter um controle sobre o consumo de brownies durante as competições e por isso precisa da sua ajuda.

Entrada

A primeira linha da entrada consiste em um número inteiro positivo, n, indicando o número de seletivas (competições) organizadas pelo professor Claudius. A seguir são dadas as entradas de cada seletiva. As primeiras entradas de uma seletiva são dois inteiros (separados por um espaço): o número de participantes (entre 1 e 30) da competição e o número de brownies (entre 60 e 600) que o professor Claudius trouxe para o *coffe-break*. A próxima linha contém um inteiro positivo, m, representando o número de quantos grupos de participantes se aproximaram da mesa de *coffe-break* para pegar brownies. Seguem então m linhas contendo um inteiro positivo cada, que representam a quantidade de participantes de cada grupo que compareceu a mesa de *coffe-break* para pegar um brownie.

Saída

Para cada seletiva, imprima uma linha contendo a frase: `"Seletiva #s:"`, onde s é o número da seletiva (começando em 1). Na sequência, para cada grupo de participantes que se aproximaram da mesa de *coffe-break*, imprima um inteiro por linha indicando o número de pedaços de brownies que restaram após cada participante do grupo que se aproximou ter pego um único pedaço de brownie. Por fim, deixe uma linha em branco após a saída de cada seletiva.

Exemplo de entrada	**Exemplo de saída**
2 20 60 8 15 10 20 18 9 12 2 10 15 100 4 1 2 3 5	Seletiva #1: 45 35 15 12 3 12 10 10 Seletiva #2: 99 97 94 89

3 Problema C: Cavalo

`Arquivo: cavalo.[c|cpp|java]` `Tempo limite: 2 s`

Pedrinho acabou de comprar um novo tabuleiro de xadrez com dimensões $N \times N$. As linhas e colunas deste tabuleiro são numeradas de `1` a `N`. Pedrinho sempre gostou muito dos movimentos dos cavalos (knight) no xadrez e está curioso: se um cavalo estiver em uma dada posição inicial do tabuleiro, na linha R_1 e coluna C_1, qual o número mínimo de movimentos necessários para alcançar a posição na linha R_2 e coluna C_2? Para tabuleiros pequenos Pedrinho até consegue calcular razoavelmente bem, mas para tabuleiros maiores ele está com dificuldades e precisa de ajuda.

Fonte: www.freevectors.net

Lembre-se, o cavalo (ou *knight*) faz movimentos no formato `L`, isto é, dada a sua posição `(A,B)` é possível alcançar as seguintes posições com um único movimento: `(A-2,B-1)`,`(A-2,B+1)`, `(A+2,B-1)`, `(A+2,B+1)`, `(A-1,B-2)`, `(A-1,B+2)`, `(A+1,B-2)` e `(A+1,B+2)`. Ah, e obviamente o cavalo não pode sair do tabuleiro.

Entrada

A primeira linha da entrada consiste em um número inteiro positivo, T, indicando o número de casos de teste. A seguir são dados cinco inteiros positivos: N ($3 \leq N \leq 20$), R_1, C_1, R_2 e C_2 ($1 \leq R_1, C_1, R_2, C_2 \leq N$).

Saída

Para cada caso de teste, imprima inicialmente `"Caso #i: "` onde i indica o número do caso de teste atual, começando em 1. Então imprima o número mínimo de movimentos que o cavalo deverá realizar para alcançar a posição (R_2,C_2) partindo de (R_1,C_1). Assuma que sempre haverá uma solução válida.

Exemplo de entrada	**Exemplo de saída**
2 5 1 1 2 3 5 1 1 2 2	Caso #1: 1 Caso #2: 4

4 Problema D: Dominos

`Arquivo: dominos.[c|cpp|java]` `Tempo limite: 2 s`

Dominós são muito divertidos, especialmente para as crianças, que gostam de alinhar as peças de pé, uma na sequência da outra (veja a figura abaixo).

Fonte: atlasofcreation.com

Quando um dominó (uma peça) é derrubado, derruba o próximo, que derruba o seguinte, até o final da linha de dominós. Entretanto, às vezes um dominó mal posicionado falha em derrubar o dominó adjacente, dando fim à sequência. Nesses casos, temos que usar nossas mãos para derrubar o próximo dominó para continuar a derrubar a linha de dominós.

Sua tarefa é determinar, dada a configuração de um conjunto de peças de dominó, o número mínimo de dominós que devem ser derrubados manualmente para que todos os dominós sejam derrubados.

Entrada

A primeira linha da entrada contém um inteiro t especificando o número de casos de teste que seguem. Cada caso de teste começa com uma linha contendo dois inteiros n e m, sendo que ambos não serão maiores que 10^5. O primeiro inteiro, n, é o número de peças de dominó e o segundo inteiro, m, é o número de linhas que seguem cada caso de teste. As peças de dominó são numeradas de 1 a n. Cada uma das linhas seguintes contêm dois inteiros x e y indicando que se o dominó x cair, fará com que o dominó y caia também. Isto não significa que o dominó y derruba o dominó x.

Saída

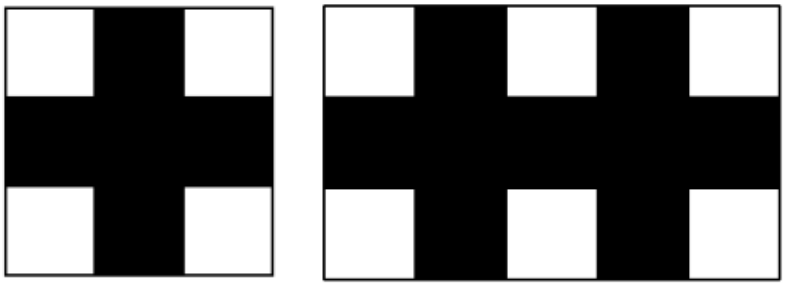
Para cada caso de teste, seu programa deve imprimir uma linha contendo um inteiro: o número mínimo de dominós que devem ser derrubados manualmente para que todos os dominós sejam derrubados.

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

5 Problema E: Estrelas

`Arquivo: estrelas.[c|cpp|java]` `Tempo limite: 2 s`

Sabe-se que a civilização maia tinha um grande interesse na observação celestial. Esse interesse inclusive pode ser visto em várias imagens encontradas nas ruínas maias. Foi descoberto recentemente que os maias utilizavam uma representação de estrela bastante simples como o principal elemento em seus desenhos, conforme ilustrado abaixo (a). As artes maias eram produzidas utilizando-se de uma combinação de estrelas, que podem inclusive se sobrepor uma sobre a outra (em partes). A imagem abaixo (b), por exemplo, poderia ser produzida com duas estrelas.



(a) Estrela simples (b) Arte composta por duas estrelas

Sabe-se que os maias não utilizavam a representação de estrela de forma inclinada, ou seja, este elemento básico de desenho sempre era pintado da forma ilustrada na figura (a). Atualmente, os pesquisadores estão interessados em descobrir qual o número mínimo de estrelas que deveriam ser pintadas para se alcançar uma dada obra. Além disso, deseja-se descobrir quais imagens não foram produzidas pelos maias, ou seja, quais imagens não poderiam ser elaboradas utilizando-se apenas de estrelas.

Entrada

A primeira linha da entrada consiste em um número inteiro positivo, T, indicando o número de imagens a serem computadas. A descrição de cada imagem inicia com uma linha contendo dois inteiros, r e c, ($1 \leq r \leq 9$, $1 \leq c \leq 9$), representando respectivamente o número de linhas e colunas da imagem. As próximas r linhas contêm c caracteres cada. Estes caracteres podem ser o símbolo '.', representando um espaço da imagem não pintado, ou o símbolo '#', representando um espaço da imagem que está coberto por tinta.

Saída

Para cada caso de teste, imprima uma resposta no formato "`Imagem #`d: s" (sem aspas) onde d indica o número da imagem (iniciando em 1) e s representa o número mínimo de estrelas que precisariam ser pintadas para representar a imagem. Caso a imagem não possa ser representada por estrelas, imprima "impossivel" (sem aspas).

Exemplo de entrada	**Exemplo de saída**
7 1 1 . 1 1 # 3 3 .#. ### .#. 3 5 .#.#. ##### .#.#. 4 7 .##.#.. ######. .###### ..#..#. 3 3 ... ... 1 2 ##	Imagem #1: 0 Imagem #2: impossivel Imagem #3: 1 Imagem #4: 2 Imagem #5: 5 Imagem #6: 0 Imagem #7: impossivel

6 Problema F: Falta Gasolina?

`Arquivo: falta_gasolina.[c|cpp|java]` `Tempo limite: 2 s`

Fonte: toonclips.com

Depois de verificar os recibos de sua viagem de carro pela Europa neste verão, você percebeu que os preços de gasolina variaram entre as cidades que visitou. Talvez você pudesse ter economizado algum dinheiro se fosse um pouco mais esperto sobre o local onde completou seu combustível?

Para ajudar outros turistas (e economizar seu próprio dinheiro da próxima vez), você quer escrever um programa para encontrar a forma mais barata de viajar entre cidades, enchendo seu tanque no caminho. Assumimos que todos os carros usam a mesma unidade de combustível por unidade de distância, e iniciam com um tanque de gasolina vazio.

Especificação da Entrada

O conjunto de entrada é formado por apenas uma instância de problema, com diversas consultas. A primeira linha de entrada é formada por $1 \leq n \leq 1000$ e $0 \leq m \leq 10000$, representando o número de cidades e estradas, respectivamente. Então segue uma linha com n inteiros $1 \leq p_i \leq 100$, onde p_i é o preço do combustível na i-ésima cidade. Então seguem m linhas com três inteiros $0 \leq u < n$, $0 \leq v < n$ e $1 \leq d \leq 100$, dizendo que há uma estrada entre u e v com distância d. Em seguida, uma linha com o número $1 \leq q \leq 100$, referente ao número de consultas, e q linhas com três inteiros $1 \leq c \leq 100$, s e e, onde c é a capacidade de combustível do veículo, s é a cidade de partida, e e é o destino.

Especificação da Saída

Para cada consulta, exiba o preço da viagem mais barata a partir de s até e, usando um carro com a capacidade dada, ou "`IMPOSSIVEL`" se não há nenhuma maneira de sair de s e chegar em e com um dado carro.

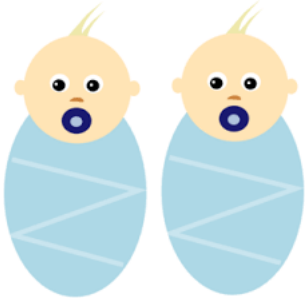
Exemplo de entrada	**Exemplo de saída**

`5 5`
`10 10 20 12 13`
`0 1 9`
`0 2 8`
`1 2 1`
`1 3 11`
`2 3 7`
`2`
`10 0 3`
`20 1 4`
	`170`
	`IMPOSSIVEL`

7 Problema G: Gemeos

`Arquivo: gemeos.[c|cpp|java]` `Tempo limite: 2 s`

Nesta olimpíada, seu Nazário tem filhos gêmeos como jogadores de futebol na equipe do Brasil. Vamos assumir que RONALDO sempre vista a camisa `18` e ROMARIO a `17`. Para ajudar o seu Nazário identificar os seus gêmeos na TV, um dispositivo foi instalado que faz uma identificação pela camisa. Este avisa quando eles estão na TV. Sim, RONALDO e ROMARIO não jogam como goleiros, pois ai é o `1`.



Fonte: www.clker.com

Problema

Dada uma lista de 10 números, determinar se os gêmeos estão lá.

Entrada

A primeira linha de entrada contém um número inteiro positivo, `N` ($N \leq 50$) indicando o número de conjuntos de dados a serem verificados. Os conjuntos são dados nas N linhas seguintes como entrada, um conjunto por linha. Cada conjunto é composto por exatamente 10 inteiros distintos separados por espaço em branco. Cada número é um inteiro entre 11 e 99 inclusive.

Saída

Como saída mprima o conjunto da entrada. Então, na próxima linha de saída, imprima uma das quatro mensagens (ROMARIO, RONALDO, NENHUM ou AMBOS), indicando quais dos gêmeos estão neste conjunto. Deixar uma linha em branco na saída após cada caso de teste. Veja os exemplos.

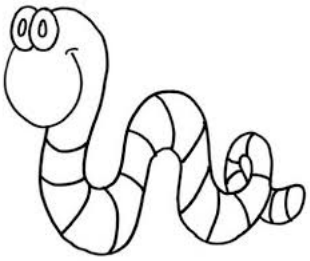
Exemplo de entrada	**Exemplo de saída**

`4`
`11 99 88 17 19 20 12 13 33 44`
`11 12 13 14 15 16 66 88 19 20`
`20 18 55 66 77 88 17 33 44 11`
`12 23 34 45 56 67 78 89 91 18`
	`11 99 88 17 19 20 12 13 33 44`
	`ROMARIO`
	
	`11 12 13 14 15 16 66 88 19 20`
	`NENHUM`
	
	`20 18 55 66 77 88 17 33 44 11`
	`AMBOS`
	
	`12 23 34 45 56 67 78 89 91 18`
	`RONALDO`

8 Problema M: Matrizes com Minhocas

`Arquivo: minhoca.[c|cpp|java]` `Tempo limite: 2 s`

Estudos científicos recentes mostraram que as minhocas provenientes de Joinville e região apresentam um comportamento inusitado ao serem colocadas sobre um tabuleiro composto por quadrados na forma de uma matriz: elas percorrem o tabuleiro, quadrado por quadrado, na forma de uma espiral. Assim, o Departamento de Catalogação de Curiosidades (DCC) da prefeitura, logo requisitou que tal comportamento fosse mapeado e modelado na forma de uma programa computacional. E nada melhor do que uma equipe de programadores da Maratona de Programação para cumprir essa tarefa. Dada uma matriz bidimensional, seu programa deve imprimir os elementos da matriz em forma de espiral, como as minhocas estudadas, tal como segue:



F.: freecoloringpages.co.uk

- imprimir os elementos da primeira linha da esquerda para a direita
- imprimir os elementos da último coluna de cima para baixo
- imprimir os elementos da última linha da direita para a esquerda
- imprimir os elementos da primeira coluna de baixo para cima
- imprimir os elementos da segunda linha da esquerda para a direita
- ....

Obs.: Cada elemento da matriz deve ser impresso apenas uma vez conforme exemplo apresentado mais abaixo.

Entrada

A entrada é dividida em conjuntos. Cada conjunto começa com dois valores inteiros, separados por um espaço, para M e N que são as dimensões da matriz (ambos os valores entre 1 e 15). Em seguida, as próximas M linhas contém os valores para as linhas da matriz. Assuma que cada valor da matriz é composto por um simples caracter e que a entrada inicia já no primeiro caracter de cada linha (nenhum espaço em branco antes da entrada), e ainda que há um espaço em branco entre cada elemento da matriz.

O fim da entrada é indicado por valores 0 e 0 para M e N.

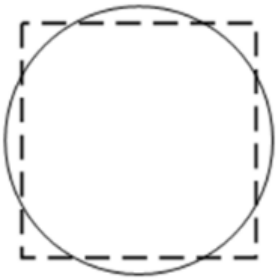
Saída

Para cada matriz, imprima a matriz original com um espaço entre os elementos. A seguir, imprima os elementos da matriz em formato "Minhoca" sem qualquer espaço entre os elementos. Deixe uma linha em branco após cada matriz de entrada. Não deixe espaços ao fim das linhas. Siga o formato apresentado no exemplo abaixo.

Exemplo de entrada	**Exemplo de saída**
3 4 A B C D J K L E I H G F 5 6 A B C D E F R S T U V G Q Z Z Z W H P Z Z Y X I O N M L K J 0 0	Matriz #1: Original: A B C D J K L E I H G F Minhoca: ABCD EF GHI J KL Matriz #2: Original: A B C D E F R S T U V G Q Z Z Z W H P Z Z Y X I O N M L K J Minhoca: ABCDEF GHIJ KLMNO PQR STUV WX YZZ Z ZZ

9 Problema P: Pluviômetro

`Arquivo: pluviometro.[c|cpp|java]` `Tempo limite: 5 s`



Como todos sabem, na cidade de Chuville chove muito. Joãozinho Snow se mudou para Chuville e está curioso para saber se realmente chove tanto assim. Para verificar isso ele vai utilizar um pluviômetro improvisado e a clarabóia da sua casa. Apesar da chuva em Chuville sempre cair de forma perpendicular ao solo (curioso não é?!), Joãozinho tem um problema: a clarabóia de sua casa tem um formato de quadrado e o seu pluviômetro tem formato circular.

Joãozinho ainda é novo e tem dificuldade para fazer cálculos matemáticos, por isso ele precisará da sua ajuda: ele precisa saber se o pluviômetro vai cobrir toda a área da clarabóia ou não. Dadas as medidas de um dos lados da clarabóia e o raio do pluviômetro, calcule a área total da clarabóia que poderá ser coberta pelo pluviômetro sendo que Joãozinho sempre vai colocar o centro do pluviômetro alinhado com o centro da clarabóia.

Entrada

A primeira linha da entrada consiste em um número inteiro positivo, T, indicando o número de casos de teste. Seguem então T linhas contendo um par de inteiros L e R ($1 \leq L \leq 100$, $1 \leq R \leq 100$), os quais representam, respectivamente, um dos lados da clarabóia e o raio do pluviômetro.

Saída

Para cada caso de teste imprima um único número real com duas casas decimais (obs: 1.234 deve ser arredondado para 1.23, ao passo que 1.235 deve ser arredondado para 1.24). Para este problema, utilize 3.14159265358979 como o valor de *pi*.

Exemplo de entrada	**Exemplo de saída**
3 1 1 8 5 10 4	1.00 62.19 50.27

10 Problema Q: Queda de Preços

`Arquivo: queda.[c|cpp|java]` `Tempo limite: 2 s`

Postos de gasolina costumam ter placas com os preços praticados (geralmente três categorias: comum, aditivada e premium). Estas placas mostram os preços praticados. O problema é que frequentemente nessas placas, alguns dígitos caem, dificultando a leitura.

Fonte: www.shutterstock.com

Problema

Dados preços para três categorias de gasolina onde, no máximo, há um dígito faltando, você deve determinar o valor exato de cada preço. Vamos assumir que a gasolina comum é mais barata que a gasolina aditivada, a qual por sua vez é mais barata que a Premium. Também assumiremos que a gasolina comum custa, no mínimo, R$ 2.00 e que a Premium custa, no máximo, R$ 5.00

Entrada

Existem múltiplos quadros de preços (casos de teste) na entrada. A primeira linha da entrada contém um número inteiro positivo n, indicando o número de quadros de preços a serem processados. Os preços estarão então nas próximas n linhas, cada um em uma linha separada. Cada quadro de preços contém três preços: o primeiro para a gasolina comum, o segundo para a aditivada e o último para a Premium. O primeiro preço começa na primeira coluna da linha, cada preço ocupa três dígitos (a entrada não contém o ponto decimal), e existe um único espaço em branco entre eles. Os caracteres utilizados em cada preço são apenas dígitos de 0 a 9 e um hífen (-) para indicar um dígito ausente (existe no máximo um dígito ausente em cada preço). Como o preço custa, no mínimo, $2.00, os dígitos 0 e 1 não irão aparecer como o primeiro caracter para um preço na entrada. Analogamente, o preço máximo ($5.00) limita o uso dos dígitos válidos como o primeiro caracter.

Saída

No início de cada caso de teste imprima "`Posto de Gasolina #g:`" onde g representa o número do caso de teste começando por 1. Para cada posto de gasolina, imprima os valores da entrada e da saída (cada um em uma linha separada e identada três colunas). Se existirem múltiplas possíveis respostas, usa o valor mínimo para a gasolina comum. A seguir, considerando esse valor mínimo da comum, calcule o valor mínimo para a gasolina aditivada. Finalmente, considere o valor mínimo da gasolina Premium com base no valor da aditivada. Assuma que os valores da entrada irão sempre resultar em, pelo menos, uma possível resposta.

Deixe um linha em branco após cada caso de teste. Siga o exemplo apresetado no exemplo abaixo. Tenha certeza de alinhar as saída utilizando espaços (e não tabs) exatamente como fornecido no exemplo.

Exemplo de entrada	**Exemplo de saída**
2 2-9 285 -99 -50 -99 -99	Posto de Gasolina #1: Entrada: 2-9 285 -99 Saida: 209 285 299 Posto de Gasolina #2: Entrada: -50 -99 -99 Saida: 250 299 399