Iracema

1^a Seletiva Interna – 2013/1

Sevidor BOCA:

`http://10.20.107.205/boca/`
(acesso interno)

`http://200.19.107.205/boca/`
(acesso externo)

Organização e Realização:

Claudio Cesar de Sá (coordenação geral), Lucas Hermann Negri (coordenação técnica), Yuri Kaszubowski Lopes, Alexandre Gonçalves Silva (revisão técnica), Roberto Silvio Ubertino Rosso Jr., André Luiz Guedes (Dinf/UFPr), Rogério Eduardo da Silva

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**;
- 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 de seu programa. 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;
- A interface KDE está disponível nas máquinas Linux, que pode ser utilizada ao invés da Unity. Para isto, basta dar *logout*, e selecionar a interface KDE. Usuário e senha: *udesc*;
- O nome *Iracema* é uma homenagem a personagem indígena de José de Alencar. Hoje, 19 de abril, dia do índio.

Patrocinador e Agradecimentos

- Linx – Patrocinador oficial do ano de 2013;
- DCC/UDESC;
- Rutes pelo empenho neste ano;
- Alguns, muitos outros anônimos.

Iracema

1ª Seletiva Interna da UDESC

19 de abril de 2013

Conteúdo

1 Problema A: Anagramas 4

2 Problema B: Beaujolais 6

3 Problema C: Corrida dos Marrecos 7

4 Problema D: Dardos 8

5 Problema E: Engarrafamento 10

6 Problema F: Quem vai à Festa? 11

7 Problema G: *Gloud Computing* 12

8 Problema H: Hexágono Perplexo 14

9 Problema I: Indefesas Estátuas 15

10 Problema J: Jogo do Maior 16

1 Problema A: Anagramas

`Arquivo: anagramas.[c|cpp|java]`

Era uma noite normal naquela ilha definitivamente nada ensolarada. Nesta mesma noite dois cientistas da computação estavam em um pub, era Saint Patrick's Day[1]. Enquanto todos cantavam, dançavam sob músicas folclóricas, bebiam Guinness, jogavam dardos, etc, os dois cientistas discutiam sobre o problema mais difícil de toda a história das maratonas de programação. *Please, one more Guinness.* Tal problema nunca havia sido solucionado e mesmo que alguém jurasse ter o algoritmo correto não seria possível conferi-lo pois nenhum juiz e nem mesmo o sujeito que bolou o problema chegou a uma solução.

Este não é o tal problema mais difícil, mas sim um problema sugerido por um dos cientistas depois de três fatores. O primeiro fator é eles já estarem de saco cheio do problema que não conseguiam resolver. O segundo fator é que eles já tinham bebido Guinness demais para resolver um problema tão complicado e eles necessitavam de um problema mais fácil para acompanhar o próximo *pint* (medida inglesa do nosso popular *choppinho*). E finalmente um deles mencionou a obra de José de Alencar, Iracema, que possuía certos anagramas. Por exemplo Iracema é anagrama de América (se desconsiderarmos o problema de acentuação). A ideia de anagramas pareceu uma boa companhia para a cerveja e então enquanto um dos cientistas foi buscar a próxima rodada, o outro formulou três questões envolvendo anagramas.

Todas as questões consideram palavras formadas por c diferentes consoantes e v diferentes vogais e as palavras tem tamanho $c+v$ (ou seja, não há letra repetida). O objetivo é calcular quantas diferentes palavras (anagramas) podem ser formadas para cada uma das seguintes restrições:

1. Todas as vogais devem estar juntas e em ordem alfabética;
2. Todas as vogais devem estar juntas em qualquer ordem;
3. As vogais não necessitam estarem juntas, mas devem aparecer em ordem alfabética na palavra.

Um dos amigos gostou tanto das questões que resolveu colocar na próxima seletiva da maratona da UDESC. Neste ponto é que você, caro maratonista, entra. Escreva um programa que a partir dos valores de `c` e `v` imprima as combinações para cada uma das restrições acima.

Entrada

A entrada é composta por vários casos de testes. Cada caso de teste é indicado por uma linha contendo dois inteiros $0 \leq c \leq 13$ e $0 \leq v \leq 5$ representando o número de consoantes e vogais na palavra respectivamente. Ainda, é garantido que $c + v > 0$. A entrada termina com $c = v = -1$.

Saída

Para cada caso de teste imprima uma linha com três inteiros, cada inteiro corresponde ao número de combinações para as restrições fornecidas.

[1] Festa anual que celebra um dos padroeiros da Irlanda, e é normalmente comemorado no dia 17 de Março pelos países que falam a língua inglesa

Exemplo de Entrada

```
3 2
1 0
0 1
-1 -1
```

Exemplo de Saída

```
24 48 60
1 1 1
1 1 1
```

2 Problema B: Beaujolais

`Arquivo: beaujolais.[c|cpp|java]`

O Beaujolais (mais conhecido como na França como *beaujolais nouveau*) é um vinho jovem que fica pronto para o consumo aproximadamente 2 meses após a colheita. A chegada do *beaujolais nouveau* é celebrado pelos franceses, sempre na terceira quinta-feira do mês de novembro. Sua chegada é anunciada em bares, supermercados, etc, com uma célebre frase: *Le Beaujolais Nouveaux est arrivée !*. Numa tradução, temos: O *beaujolais* novo chegou!

Aproveitando a sua viagem enóloga pela França, o professor Omerix trouxe algumas garrafas vinhos tais como: beaujolais, tintos de Bordeaux, *terroir* da Auvergne, os *nobres* da Bourgonha, *rose* da Provence, etc. Vinhos de diversas regiões e características. Ao chegar no Brasil, Omerix se deparou que cada vinho tinha uma temperatura mínima e máxima para sua conservação ideal. Sua tarefa é ajudar o professor Omerix a escolher a temperatura ideal para sua secreta adega (deixou de ser secreta), de forma a cobrir uma faixa com a maior quantidade possível de vinhos. Claro, o prof. Omerix está preocupado com sua conta de energia elétrica e a sustentabilidade do planeta.

Entrada

A entrada é composta por vários casos de testes. Cada caso de teste é iniciado por um linha com um valor `n` que indica a quantidade de vinhos a serem avaliados. Cada uma das n linhas seguintes é composta por dois valores: t_{min} e t_{max}, a temperatura mínima e a máxima para uma boa conservação do vinho. Sabe-se que a adega do professor Omerix aceita temperaturas inteiras que seguem de `-10` graus até `20` graus Celsius, pois atende aos vinhos que não dão dor de cabeça no dia seguinte. Resumindo: $-10 \leq t_{min}$ e $t_{max} \leq 20$ e $1 \leq n \leq 1000$. A entrada termina com `n = 0`.

Saída

Para cada caso de teste imprima uma linha com um inteiro que represente a maior temperatura que contemple o maior números de vinhos. Caso não exista uma temperatura comum que esteja presente em dois ou mais vinhos, imprima a imprima a temperatura t_{max} do vinho que tenha o menor valor de t_{min} (uma chance de preservar os vinhos por temperatura baixa, dado o nosso clima tropical). Veja os exemplos dos casos de testes. Para `n = 0` não imprima nada.

Exemplo de Entrada

```
3
-3 5
-1 10
9 15
2
3 5
7 15
5
1 3
2 4
3 5
4 6
1 10
0
```

Exemplo de Saída

```
10
5
4
```

3 Problema C: Corrida dos Marrecos

`Arquivo: corrida.[c|cpp|java]`

Pirabeiraba é um distrito de Joinville, onde colonizadores alemães se instalaram no início do século XX. Anualmente há a festa do aipim, tubérculo conhecido como macaxeira no nordeste do Brasil. Para acompanhar o aipim, nada como um prato típico germânico: o marreco recheado! Para os entendidos de culinária, há uma magia nesta combinação: marreco com aipim. Contudo, para matar o marreco, você deve capturá-lo quando este estiver com o sangue bem quente. Para isto, o marreco deve estar cansado. Dizem que seu sangue quente é sinônimo de fertilidade, para não dizer: afrodisíaco! Mas isto é uma outra história.

Nesta brincadeira de correr atrás do marreco, surgiu a idéia de cansá-los com uma corrida entre eles. O espaço físico da Sociedade Rio da Prata é limitado, assim, construíram apenas 3 raias para se realizar estas corridas. As corridas são feitas em grupos de 2 e 3 marrecos. Os primeiros colocados destes grupos são novamente divididos em grupos de 2 ou 3 para uma nova rodada. Isto acontece até que só reste o marreco campeão, que, como prêmio foge (por ora) da panela. Todos os marrecos *sobreviventes* devem correr na rodada, isto é, se não for possível dividir todos os marrecos em grupos de 3, alguns grupos de 2 devem ser formados, mas de forma a minimizar o número de corridas. Exemplos são vistos na Figura 1.

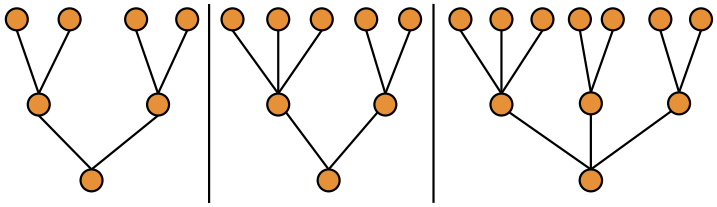

Figura 1: Exemplos: competição com 4, 5 e 7 marrecos.

Os marrecos perdedores, por sua vez, serão os primeiros a irem para panela. Você foi convidado para comer marreco com aipim, mas, em troca, deve escrever um programa que calcule o número de corridas realizadas para se determinar o marreco campeão.

Entrada

A entrada do programa é composta por vários casos de teste. Cada caso de teste é composto por uma linha contendo um número `n` (número de marrecos), tal que $1 \leq n \leq 100.000$, e `n = 0` é utilizado unicamente para marcar o término das entradas, sendo que este deve ser desconsiderado.

Saída

O seu programa deve imprimir na saída padrão uma linha por caso de teste, contendo o número de corridas necessárias para escolher o marreco campeão.

Exemplo de Entrada

```
3
4
5
6
7
0
```

Exemplo de Saída

```
1
3
3
3
4
```

4 Problema D: Dardos

`Arquivo: dardos.[c|cpp|java]`

Ainda lá na Sociedade Rio da Prata, existe a festa anual do colono todo dia 25 de Julho. Uma festa de 3 a 4 dias para alemão algum por defeito. Entre a corrida do marreco, comida e música alemã, concurso do tomador de chopp, serrador de lenha, tiro de estilingue, corrida do saco, corrida do tamanco, concurso de descascadores de aipim, etc, existe o campeonato de tiro ao alvo.

Este tiro ao alvo é com dardos, a fim de manter tudo ecologicamente correto, e reusá-los. Com o passar dos anos, a festa evoluiu, e um sistema automático de contar os pontos foi construído.

O construtor do alvo era um alemão esperto, fez um alvo (visto na Figura 2) e instalou sensores capazes de ler a posição de cada dardo com precisão. Contudo, o alemão esqueceu de fazer o sistema que soma os pontos do jogador. Claro, esta nobre tarefa é para os criativos maratonistas.

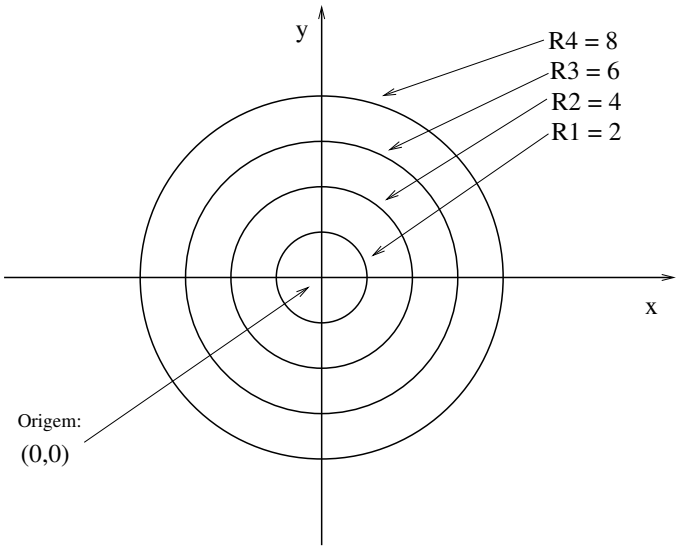

Figura 2: Alvo do jogo de dardos.

No alvo encontram-se circunferências de diferentes raios, todas centradas no mesmo ponto $(0, 0)$, formando assim anéis concêntricos. A regra de pontuação estabelecida é em função do anel em que cada dardo acertou, seguindo a Tabela 1. Caso um dardo fique exatamente na divisa entre dois anéis, considere o anel de maior pontuação.

	Raio	Pontuação
$R1$	2	7 pontos
$R2$	4	5 pontos
$R3$	6	3 pontos
$R4$	8	1 pontos

Tabela 1: Pontuação por anel atingido.

Sua tarefa é computar o somatório de pontos de cada jogador/atirador, a partir de uma lista das posições finais de cada dardo arremessado no alvo.

Entrada

A entrada é composta por vários casos de teste. A primeira linha consiste de um número n, que corresponde a quantidade de tiros a serem avaliados de um atirador. As n linhas seguintes são dadas por um par de coordenadas x_i y_i que corresponde as coordenadas do dardo no alvo para o i-ésimo arremesso. Os valores de x_i e y_i tem precisão de centésimos. A entrada é encerrada com `0`, isto é, não há mais atiradores.

Saída

O programa deve imprimir em uma linha o total de pontos do atirador. Neste linha deve constar o somatório dos pontos do referido atirador.

Exemplo de Entrada

```
2
0 2
1.75 1.75
4
-1 0.5
6.5 0.2
2 4.5
2 -2
0
```

Exemplo de Saída

```
12
16
```

5 Problema E: Engarrafamento

`Arquivo: engarrafamento.[c|cpp|java]`

Marcos é um cientista da computação que trabalha em uma empresa de transporte, analisando dados das viagens dos caminhões de carga da empresa e otimizando rotas de veículos. Devido aos constantes congestionamentos envolvendo veículos da empresa, esta designou uma nova tarefa para Marcos: computar a distância percorrida por cada veículo em situações de tráfego intenso. Marcos está muito ocupado com outras tarefas na empresa, e designou esta tarefa de calcular a distância percorrida em viagens para você, o novo estagiário da empresa.

De forma mais específica, a distância percorrida de cada viagem deve ser calculada a partir dos dados de aceleração durante a viagem. Nestes dados constam as faixas de tempo onde o motorista pisou no acelerador (aceleração constante de $1m/s^2$, até uma velocidade máxima de $10m/s$), sendo que quando o veículo não estiver acelerando o motorista estará com o pé no freio (veículo parado ou freando com uma desaceleração constante de $2m/s^2$). A distância total percorrida na viagem deve ser calculada a partir destas faixas de tempo, assumindo que o veículo está inicialmente parado.

Entrada

A entrada é composta por vários casos de teste. Cada caso corresponde a uma viagem e é iniciado por uma linha contendo um inteiro N, que diz a quantidade de faixas de aceleração do veículo durante a viagem. O final da entrada é marcado com $N = 0$, caso que não deve ser processado.

Cada uma das próximas N linhas contém dois inteiros, a e b, designando as faixas de tempo (em segundos) onde o motorista está com o pé no acelerador (acelerou no tempo $t = a$ até $t = b$). No primeiro caso do exemplo abaixo (primeira viagem) o motorista pisou no acelerador no tempo $t = 0$s até $t = 5$s, pisou no freio entre $t = 5$s e $t = 8$s, acelerou de $t = 8$s até $t = 15$s, freou entre $t = 16$s e $t = 17$s e acelerou até $t = 50$s. A distância percorrida deve ser computada de $t = 0$s até o segundo final da última faixa de aceleração, neste caso de $t = 0$s até $t = 50$s.

Limites: $0 \leq N \leq 1000$, sendo que cada viagem demora no máximo 30h.

Saída

O programa deverá imprimir, para cada viagem, uma linha contendo a distância percorrida em metros (com duas casas decimais).

Exemplo de Entrada

```
3
0 5
8 15
17 50
1
5 30
0
```

Exemplo de Saída

```
358.75
200.00
```

6 Problema F: Quem vai à Festa?

`Arquivo: festa.[c|cpp|java]`

A garoa fina que caiu na última semana levou o professor Claudius Virux ficar saudoso de seus tempos de Campina Grande – Pb, como estudante na UFCG. Lá, o inverno é um período marcado por uma *chuvinha* igual a de Joinville, com noites frias.

Na universidade, as *festinhas* dos estudantes, que ocorriam em quase todas as sextas-feiras e sábados, eram um programa e tanto. O mais interessante é que se encontravam as *figuras* mais inusitadas, poetas, cantores, professores, e outros artistas, além, é claro, dos estudantes. Sempre havia algum motivo para organizar uma festa, quando não, a ideia era celebrar a semana dura da universidade que se passou.

O mais curioso era a sistemática de como o anfitrião fazia o convite. O dono(a) da casa (em seus tempos se chamava de *república*) convidada os seus amigos imediatos, estes por sua vez convidavam outros, e assim, sucessivamente. Chegado o dia da festa, o anfitrião queria conhecer os seus novos amigos, a fim de verificar como tal *corrente* de convite tinha se propagada.

Para controlar quantos e como os convidados vieram à festa, o anfitrião solicitou a cada um que chegasse, escrevesse seu nome, e quem o convidou.

Sua tarefa é contar quantos convidados estão presentes em cada festa, dada apenas a relação imediata entre convidado e um amigo.

Entrada

Para cada festa, haverá um número de relação entre os convidados. Este é um valor n escrito antes das relações que seguem em pares do tipo $(x, y) = (y, x)$. Onde x é o nome de um amigo e y o seu convidado. Os valores de x e y são numerados de 1 a 1000, e **o anfitrião é sempre o número** 1.

A leitura de 0 na entrada indica fim de festa! O formato dessas entradas seguem os padrões abaixo.

Saída

A cada conjunto de relação, imprima o número total de participantes na festa, incluindo o anfitrião. O total da festa, uma por linha. Na entrada 0 não escreva nada.

Exemplo de Entrada

```
3
(1,2) (2,3) (4,5)
3
(2,3) (3,4) (4,5)
5
(1,2) (5,2) (6,5) (5,4) (4,3)
0
```

Exemplo de Saída

```
3
1
6
```

PS: No caso 2, o anfitrião não convidou ninguém. Brigou com a *paquera* (esta era gíria daquela época) e ficou bebendo sozinho em casa, *na sua festa particular*.

7 Problema G: *Gloud Computing*

`Arquivo: gloud.[c|cpp|java]`

A *Gloud Computing* está vindo se instalar para região de Joinville. Eles são conhecidos por proverem aplicativos na internet, mais especificamente um modelo de negócio baseado em *cloud computing* – computação nas nuvens.

A fim de selecionar os novos funcionários da empresa, eles contactaram o comitê da maratona da UDESC, para que passassem um problema aos nossos maratonistas. Aquele que resolver, além do balão, pode preencher a ficha funcional com *estrelinhas* a mais.

Basicamente, a *Gloud Computing* tem aplicações espalhadas em seus servidores em diversos lugares do mundo. Estes servidores são especializados em uma lista de aplicativos a serem usados pelos usuários ali conectados na internet das nuvens.

Por exemplo, o servidor de Joinville pode disponibilizar a aplicação a, enquanto que o de Pasadena na Califónia provê as aplicações a, b e c e o servidor de Pomerode provê a aplicação c.

Temos um conjunto de servidores e cada um com um conjunto de aplicações a serem disponibilizadas a um conjunto de usuários. Cada usuário pode estar conectado a um ou mais servidores dependendo de sua demanda, como ilustrado na figura 3.

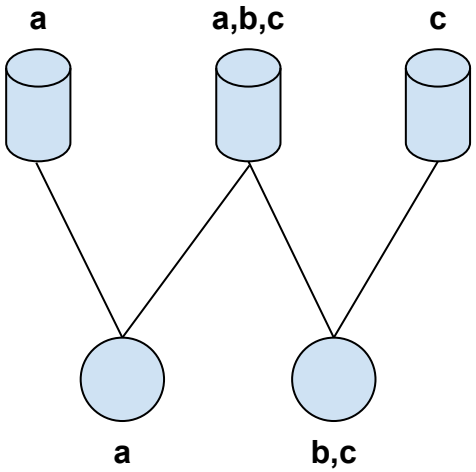

Figura 3: 3 provedores de serviços, 2 usuários e 4 conexões.

Serão disponibilizados a você dados sobre destes dois conjuntos, servidores e demanda dos usuários, e você deverá dizer a quantidade total de conexões entre clientes e servidores. As conexões são feitas de forma a maximizar a redundância. Por exemplo, se um cliente quer utilizar as aplicações b e c, ele irá se conectar a todos os servidores que disponibilizarem ao menos a aplicação b e a todos os que disponibilizarem ao menos a c. Múltiplas conexões entre um mesmo par de cliente e servidor são contabilizadas como uma só. Pode ser que um cliente requeira uma aplicação inexistente, assim como o caso de um servidor prover uma aplicação não requisitada por nenhum cliente.

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é iniciado por dois inteiros, N e M, que correspondem ao número de servidores e ao número de clientes. Cada uma das próximas N linhas contém um valor Q_i correspondente ao número de aplicações

fornecidas pelo i-ésimo servidor, seguido por Q_i palavras (separadas por espaços) referentes aos nomes das aplicações fornecidas. Após esta descrição dos servidores, seguem M linhas, cada uma contendo um valor P_j correspondente ao número de aplicações requisitadas pelo j-ésimo cliente, seguido por P_j palavras (separadas por espaços) referentes aos nomes das aplicações requisitadas. A entrada termina quando $N = M = 0$.

Limites: $0 \leq N, M \leq 200$, $0 \leq Q_i, P_j \leq 100$. Todos os nomes de aplicativos tem tamanho entre 1 e 20 caracteres. Estes nomes são formados por caracteres minúsculos e números justapostos, isto é, sem espaço em branco.

Saída

Para cada caso de teste, o programa deve imprimir a soma total de conexões entre clientes e servidores em uma linha, desconsiderando múltiplas conexões entre um mesmo par de cliente e servidor.

Exemplo de Entrada

```
3 2
1 a
3 a b c
1 c
1 a
2 b c
5 2
2 s1 s2
2 s3 s4
2 s5 s6
2 s7 s8
2 s1 s2
3 s1 s2 s20
3 s1 s2 s21
0 0
```

Exemplo de Saída

```
4
4
```

8 Problema H: Hexágono Perplexo

`Arquivo: hexagono.[c|cpp|java]`

Um quebra-cabeças bem conhecido consiste em 7 peças hexagonais com números de 1 até 6 escritos nos seus lados. Cada peça tem um arranjo diferente dos números nos seus lados. O objetivo é colocar as 7 peças em um arranjo como o mostrado na Figura 4. De forma que os números em cada aresta compartilhada, sejam idênticos. A primeira ilustração na Figura 4 à esquerda (a), é um exemplo de solução.

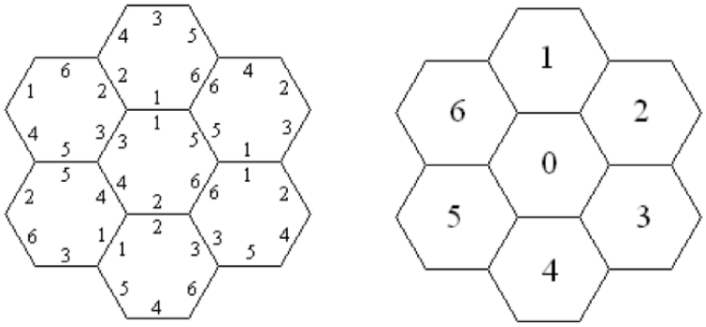

Figura 4: (a)–Uma solução exemplo (b)–Notação da posição na saída

Qualquer solução quando rotacionada, também dá uma outra solução trivialmente idêntica. Para evitar esta redundância, nós vamos lidar apenas com soluções que tenham **o número `1` na aresta superior da peça central**, como no exemplo da figura.

Entrada

A primeira linha do arquivo de entrada irá conter um único número inteiro que indica o número de casos de teste. Cada caso, será constituído por uma única linha contendo `42` inteiros. Os primeiros `6` números representam os valores na peça `0` enumerados no sentido horário, a segunda sequência de `6` números representa os valores na peça `1`, e assim por diante. Veja a ilustração (b) da figura 4.

Saída

Para cada caso de teste, imprima o número do caso (usando o formato mostrado abaixo) seguido pela frase `Sem solucao` ou pela especificação de uma solução. Uma especificação de solução lista os números das peças na ordem mostrada na Notação de Posição da figura (b). Assim, se a peça 3 está no centro, um 3 é impresso primeiro. Se a peça 0 está no topo, um 0 é impresso em segundo e assim por diante. Para cada caso de teste é garantido que há no máximo uma solução.

Exemplo de Entrada

```
2
3 5 6 1 2 4 5 1 2 3 6 4 2 3 5 4 1 6 3 1 5 6 2 4 5 4 1 3 6 2 4 2 3 1 5 6 3 6 1 2 4 5
6 3 4 1 2 5 6 4 3 2 5 1 6 5 3 2 4 1 5 4 6 3 2 1 2 5 6 1 4 3 4 6 3 5 2 1 1 3 5 2 6 4
```

Exemplo de Saída

```
Caso 1: 3 0 5 6 1 4 2
Caso 2: Sem solucao
```

9 Problema I: Indefesas Estátuas

`Arquivo: indefesas.[c|cpp|java]`

Felix João Humilde é um *nerd* rico residente em uma cidade pequena do Centro-Oeste. Ele erigiu quatro estátuas em sua homenagem no parque da cidade (que por acaso pertence a ele). Felix é muito orgulhoso das estátuas. Mas agora está preocupado com vândalos, crianças pequenas com chicletes e com os cães com infecções urinárias. Para resolver esse problema, ele decidiu construir uma cerca em volta das estátuas (Felix também é proprietário da empresa local de construção de cercas). Por várias razões estéticas ele gostaria de ter as seguintes condições atendidas:

1. O espaço fechado tem de ser um quadrado;
2. A distância entre cada estátua e seu lado mais próximo da cerca de vedação deve ser de 5 pés;
3. Duas estátuas não devem ter o mesmo lado da cerca como o mais próximo.

Depois de trabalhar por um total de 12 segundos, Felix percebeu que ele não tinha a menor idéia do comprimento de cerca necessário. Nem mesmo se as condições acima podem ser satisfeitas. Desde que ele está planejando homenagens similares em outros parques que possui, ele gostaria que alguém escrevesse um programa para resolver este problema (que mais tarde Felix irá tomar o crédito para si, é claro). Obs.: O pé (*foot*) é uma medida de comprimento do sistema imperial britânico e equivale a 30.48 centímetros.

Entrada

A primeira linha do arquivo de entrada irá conter um inteiro `n` indicando o número de casos de teste. Os casos de teste vem em seguida, um por linha, consistindo cada um de oito valores inteiros que são as coordenadas `x` e `y` da primeira, segunda, terceira e quarta estátua. Todos os valores são em pés e ficarão entre `-100` e `100` (valores inteiros). Duas estátuas não podem estar na mesma localização.

Saída

Para cada caso de teste imprima o número do caso seguido de uma das duas respostas:

1. Se tiver uma solução, imprima o comprimento do lado do quadrado que contém as estátuas. Se existirem múltiplas soluções, imprima o comprimento do lado da solução de maior tamanho.
2. Se não for possível construir uma cerca que cumpra as exigências de Felix, imprima:
   `Caso ...: Sem solucao`
   Todos os resultados numéricos impressos devem ser arredondados para o centésimo de pé mais próximo. Siga os formatos dos exemplos abaixo.

Exemplo de Entrada

```
4
0 1 2 5 5 4 3 0
10 10 30 30 20 20 0 10
0 1 1 0 3 4 4 2
0 1 0 2 0 3 0 4
```

Exemplo de Saída

```
Caso 1: 15.00
Caso 2: Sem solucao
Caso 3: 14.00
Caso 4: Sem solucao
```

10 Problema J: Jogo do Maior

`Arquivo: jogo.[c|cpp|java]`

Ogg gosta muito de brincar com seus filhos. Seu jogo preferido é o *jogo do maior*, de autoria própria. Este passatempo (no tempo das cavernas se tinha muito tempo disponível para jogos) é jogado em dupla, Ogg e um dos seus filhos. O jogo procede da seguinte forma: os dois participantes escolhem um número de rodadas e, a cada rodada, cada participante diz um número de 1 até 9 em voz alta, sendo que o participante que falar o número mais alto ganha um ponto (em caso de empate, ninguém ganha o ponto). No final das rodadas, os pontos são contabilizados e o participante com o maior número de pontos ganha. Ogg e seus filhos gostam muito do jogo, mas se perdem na contagem dos pontos. Você conseguirá ajudar Ogg a verificar a pontuação de uma lista de jogos?

Entrada

A entrada é composta por vários casos de teste (partidas). Cada caso é iniciado com um inteiro N (de 0 até 10) representando o número de rodadas da partida, sendo que o valor 0 representa o final da entrada e não deve ser processado. Cada uma das próximas N linhas contém dois inteiros, A e B, onde A é o número escolhido pelo primeiro jogador e B é o número escolhido pelo segundo jogador ($0 \leq A, B \leq 10$).

Saída

A saída deve ser composta por uma linha por caso de teste, contendo o número de pontos de cada jogador, separados por um espaço.

Exemplo de Entrada

```
3
5 3
8 2
5 6
2
5 5
0 0
0
```

Exemplo de Saída

```
2 1
0 0
```