Universidade do Estado Santa Catarina (UDESC)

Universidade Federal do Paraná (UFPr)

Maratona Doméstica de Programação – 2009 – 2

(Prova Oficial – Jennifer)

`http://200.19.107.25:8001/boca`

Organizadores: Claudio (DCC), Rosso (DCC), André Guedes (DInf)

29 de agosto de 2009

Lembrete:

- É permitido consultar livros, anotações ou qualquer outro material impresso cópia durante a prova.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
- Todos os problemas têm o mesmo valor na correção. Logo, leia e comece com os mais fáceis. Para esta prova, há vários fáceis.
- Procure resolver o problema de maneira eficiente. Na correção, a eficiência também é levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Utilize o *Clarification* para dúvidas da prova. Opcionalmente, os juízes respondem, contudo, os esclarecimentos são acessíveis a todos.
- Responsável pela construção da prova: Claudio – UDESC
- Revisão Técnica: André – UFPR

Agradecimentos

- Ao pessoal do suporte da UDESC, do DCC, Grupo Colméia,
- Pelas traduções: Fernando Deeke Sasse (DMAT), Alexandre (DCC), Adriano (DCC), Dênio (DCC), Gilmário (DCC), Daniela (DCC).
- Ao patrocínio da Loan Solutions.
- Ao DCC e demais anônimos que tornaram este evento uma realidade.

Sumário

1 Problema A: Pintura em Preto e Branco **3**

2 Problema B: Permutações Ambíguas **4**

3 Problema C: Converter Quilômetros em Milhas **5**

4 Problema D: *Todos em Todos* **7**

5 Problema E: Ruas às Escuras **8**

6 Problema F: Rede Ótica **9**

7 Problema G: Fazendo Cilindros **11**

8 Problema H: Gera Números Aleatórios **12**

9 Problema I: Quem Vai Ser Reprovado? **13**

10 Problema J: Decorando a Parede **14**

1 Problema A: Pintura em Preto e Branco

Introduçao

Você está visitando o MASP (Museu de Arte de SP) que contém uma grande quantidade de pintura moderna. Você nota, particularmente, um quadro que consiste somente de quadrados pretos e brancos, arranjados em linhas e colunas como em um tabuleiro de xadrez (dois quadrados adjacentes não tem a mesma cor).

Uma vez que você está entediado, mas lembra saudosamente, que os professores da UDESC mencionavam dos vários problemas de tabuleiros de duas cores que podiam serem montados.

Você se lembra do tabuleiro de xadrez, e pergunta: *quantos tabuleiros de xadrez de* 8 *linhas por* 8 *colunas estão embutidos daquele quadro*. O canto inferior direito do tabuleiro de xadrez deve ser sempre branco.

Especificação da Entrada e Saída

A entrada contém alguns testes. Cada teste consiste de uma linha com 3 inteiros n, m e c ($8 \leq n, m \leq 40000$), onde n é o número de linhas do quadro, e m é o número de colunas do quadro. Quanto ao c será 0 ou 1, onde 0 indica que o canto inferior direito é preto, e 1 indica que é branco.

O último teste é seguido por uma linha contendo 3 zeros.

Para cada teste, imprima o número de tabuleiros de xadrez embutidos no quadro dado.

Exemplo de Entrada	**Exemplo de Saída**

`8 8 0`
`8 8 1`
`9 9 1`
`40000 39999 0`
`0 0 0`
	`0`
	`1`
	`2`
	`799700028`

2 Problema B: Permutações Ambíguas

Alguns problemas de competições de programação são realmente complicados: não somente por exigirem um formato de saída diferente do que você poderia ter esperado, mas também pelo exemplo de resultado não mostrar a diferença. Por exemplo, vamos olhar as permutações. Uma **permutação** dos inteiros de 1 a n é uma ordenação destes inteiros. Então, o caminho natural para representar uma permutação é listar os inteiros nesta ordem. Com $n = 5$, uma permutação poderia ser algo como $2, 3, 4, 5, 1$.

No entanto, existe uma outra possibilidade de representar uma permutação: Você cria uma lista de números, onde o i-ésimo número é a posição do inteiro i na permutação. Vamos chamar esta segunda possibilidade de uma **permutação inversa**. A permutação inversa para a sequência acima é de $5, 1, 2, 3, 4$.

Uma **permutação ambígua** é uma permutação que não pode ser distinguida de sua permutação inversa. A permutação $1, 4, 3, 2$, por exemplo, é ambígua, porque sua permutação inversa é a mesma. Para evitar esta incômoda amostra de casos de teste, você deverá escrever um programa que detecta se uma determinada permutação é ambígua ou não.

Especificação da Entrada

A entrada contém vários casos de teste.

A primeira linha de cada caso de teste contém um inteiro n ($1 \leq n \leq 100000$). Então uma permutação dos inteiros 1 a n segue na próxima linha. Há exatamente um caracter de espaço entre inteiros consecutivos. Você pode assumir que todo inteiro entre 1 e n aparece exatamente uma vez na permutação.

O último caso de teste é marcado por um zero.

Especificação de Saída

Para cada caso de teste, retorne se a permutação é ambígua ou não. Utilize o formato indicado no exemplo de saída.

Exemplo de entrada	Exemplo de saída
4 1 4 3 2 5 2 3 4 5 1 1 1 0	ambiguous not ambiguous ambiguous

3 Problema C: Converter Quilômetros em Milhas

Introdução

Este ano, Força Bruta passou as suas férias em Joinville, Santa Catarina, onde ele quer treinar para a sua próxima meia maratona (corrida com mais de 21 km). No seu primeiro treino ele correu até a casa do seu amigo Guloso da Silva, que esta há 21 milhas distante de Joinville.

Chegando lá, ele está muito cansado e percebe que 21 milhas são muito mais do que 21 km. Guloso da Silva diz a ele que 21 km é igual a 13 milhas. 21 ou 13? Força Bruta percebe imediatamente que deve haver uma relação mais profunda! Pois, ambos, 13 e 21 são números de Fibonacci!

Números de Fibonacci podem ser definidos como segue:

- $F_1 = 1$
- $F_2 = 2$
- $F_{n+1} = F_n + F_{n-1} para\ n > 1$

Força Bruta acabou de estudar sobre o sistema de números de Fibonacci em sua universidade. Cada inteiro positivo x pode ser escrito como a soma de diferentes números de Fibonacci, isto significa que existe números k e $b_1, b_2, ..., b_k$ tal que $x = \Sigma_{i=1..k} b_i \times F_i$, onde $b_k = 1$ e $b_i\ (1 < i < k)$ é tanto 0 ou 1. Em suma, podemos escrever a representação como: $b(x) = (b_k, b_{k-1}, ..., b_1)$. Para tornar a representação única, solicitamos que $b_i \times b_{i-1} = 0$ para todo $i > 1$.

Por exemplo, 21 pode ser representado como $(1, 0, 0, 0, 0, 0, 0)$ e 13 como $(1, 0, 0, 0, 0, 0)$ no sistema de Fibonacci. Força Bruta informa que se pode converter uma distância x em Km em uma distância correspondente y em milhas, usando os seguintes passos:

1. Primeiro, escrever x no sistema de representação de Fibonacci $b(x)$.
2. Segundo, deslocar os bits de $b(x)$ uma posição para a direita (o último bit é eliminado) e obtenha $b(y)$.
3. Terceiro, calcular y a partir de $b(y)$, pela valoração da soma indicada acima.

Por exemplo, o número 42 escrito no sistema de Fibonacci é $(1, 0, 0, 1, 0, 0, 0, 0)$. Na segunda etapa, deslocamos os bits uma posição para a direita e obtemos $(1, 0, 0, 1, 0, 0, 0)$. Na terceira etapa, calculamos $0 * 1 + 0 * 2 + 0 * 3 + 1 * 5 + 0 * 8 + 0 * 13 + 1 * 21 = 26$.

Agora é sua vez de escrever um programa que converta quilômetros em milhas, de acordo com o algoritmo apresentado por Força Bruta a Guloso da Silva.

Especificação da Entrada

A primeira linha da entrada contém t, o número de distâncias. Força Bruta pretende converter de Km para milhas, onde t é $(0 < t < 25000)$. Cada uma das próximas linhas contém um inteiro com a distância x tal que $(2 < x < 25000)$ em quilômetros.

Especificação da Saída

Para cada distância x em quilômetros, apresente a saída y em milhas, calculadas de acordo com o algoritmo apresentado por Força Bruta.

Exemplo de Entrada	Exemplo de Saída
5 42 100 180 300 360	26 62 111 185 222

4 Problema D: *Todos em Todos*

Introdução

Você elaborou uma nova técnica de criptografia que codifica uma mensagem inserindo entre os seus caracteres strings gerados aleatoriamente de uma forma inteligente. Devido a questões de patentes pendentes não iremos discutir em detalhes a forma como as strings são gerados e inseridas na mensagem original. Porém, para validar o método, é necessário escrever um programa que verifica se a mensagem foi realmente codificada na string final.

Assim, dadas duas strings **s** e **t**, você terá que decidir se **s** é um substring de **t**, isto é, se você pode remover caracteres de **t** de forma que a concatenação do caracteres restantes de **t** seja **s**.

Especificação de Entrada

A entrada contém vários casos. Cada um é especificado por dois strings de caracteres alfanuméricos ASCII **s** e **t** separados por espaço em branco. A entrada é terminada por EOF.

Especificação de Saída

Para cada caso da entrada, `Yes` se **s** é um substring de **t** ou `No`, caso contrário.

Exemplo de Entrada	**Exemplo de Saída**

`sequence subsequence`
`person compression`
`VERDI vivaVittorioEmanueleReDiItalia`
`caseDoesMatter CaseDoesMatter`
	`Yes`
	`No`
	`Yes`
	`No`

5 Problema E: Ruas às Escuras

Introdução

A crise econômica está em toda parte, até mesmo em Baitelândia, então, para reduzir os custos operacionais o governo decidiu otimizar a iluminação das ruas. Até agora todas as ruas eram iluminadas a noite inteira, o que custava 1 dólar Baitelandês por metro por dia. Para economizar, eles decidiram não mais iluminar todas as ruas desligando a iluminação pública de algumas delas, e para ter certeza de que os habitantes ainda se sintam seguros, pelo menos um caminho entre cada par de cruzamentos deve permanecer aceso.

Mantendo a seus habitantes seguros, qual é a máxima economia diária que o governo de Baitelândia terá?

Especificação da Entrada de Dados

O arquivo de entrada contém alguns casos para teste, cada caso começa com dois números *m* (quantidade de cruzamentos) e *n* (quantidade de ruas), $1 \leq m \leq 200000$ e $m - 1 \leq n \leq 200000$, onde `m=n=0` marcam o final de arquivo.

Cada uma das *n* triplas de inteiros x, y, z especificam que haverá uma rua de mão dupla entre x e y com comprimento z metros $(0 \leq x, y < m$ e $x \neq y)$. O grafo especificado é conexo e o comprimento total de todas as ruas em cada caso de teste é menor do que 2^{31}.

Especificação da Saída

Para cada caso de teste imprima um alinha contendo a máxima economia diária que o governo pode conseguir.

Exemplo de Entrada	**Exemplo de Saída**

`7 11`
`0 1 7`
`0 3 5`
`1 2 8`
`1 3 9`
`1 4 7`
`2 4 5`
`3 4 15`
`3 5 6`
`4 5 8`
`4 6 9`
`5 6 11`
`0 0`
	`51`

6 Problema F: Rede Ótica

Pegando a onda da Internet, várias empresas emergentes decidiram construir uma rede ainda melhor, chamada de "*OticaNet*". Eles já tinham instalados vários nós que atuam como roteadores em todo o mundo. Infelizmente, elas começaram a se desentender sobre as conexões das linhas, assim, essas empresas decidiram colocar o seu próprio conjunto de cabos entre alguns dos nós. Agora, os usuários dos serviços de telefonia, Internet, etc, querem saber de como enviar os dados do nó A ao nó B, e qual é empresa capaz de proporcionar as ligações necessárias. Ajude os usuários, respondendo as suas perguntas.

Especificação da Entrada

A entrada contém vários casos testes. Cada caso de teste começa com o número de nós (nodos) da rede n. A entrada é finalizada com $n = 0$. Caso contrário, $1 <= n <= 200$. Os nós são numerados de $1, ..., n$. Depois, segue uma lista de ligações/conexões. Cada ligação começa com dois números A e B.

A lista de conexões é finalizada com $A = B = 0$. Caso contrário, $1 <= A$ e $B <= n$, e eles indicam o início e o final de uma conexão unidirecional, respectivamente. Para cada conexão, os dois nós (nodos) são seguidos pelas empresas que têm uma conexão/ligação do nó A ao nó B. A empresa é identificada por **uma** letra minúscula. Logo, o conjunto de empresas com uma mesma ligação é apenas uma palavra composta de letras minúsculas.

Após a lista de conexões/ligações de cada caso teste, esse é complementado por uma lista de perguntas. Cada consulta consiste de dois números A e B. A lista (e com ele o teste caso) é finalizado por $A = B = 0$. Caso contrário, $1 <= A$ e $B <= n$, e eles indicam o início e o final da consulta. Você pode assumir que nenhuma ligação/conexão e consulta contenha os mesmos nós de início e fim.

Especificação da Saída

Para cada consulta/pergunta em cada caso teste, gere uma linha contendo os identificadores de todas as empresas, que podem encaminhar pacotes de dados sobre as suas próprias ligações a partir do nó início ao nó fim da consulta. Se não existirem empresas que ofereça tal conexão, imprima - como saída. Após cada caso teste imprima uma linha em branco.

Exemplo de Entrada	Exemplo de Saída
3 1 2 abc 2 3 ad 1 3 b 3 1 de 0 0 1 3 2 1 3 2 0 0 2 1 2 z 0 0 1 2 2 1 0 0 0	ab d - z -

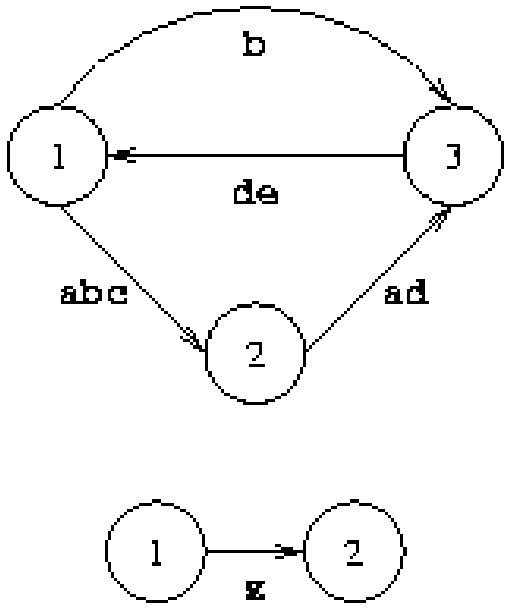

Figura 1: Grafo que ilustra os exemplos.

7 Problema G: Fazendo Cilindros

Introdução

Usando uma folha de papel e tesoura, você pode cortar as duas faces para formar um cilindro da seguinte forma:

1. Cortar o papel na horizontal (paralelo ao lado menor) para obter duas peças retangulares.
2. A partir da primeira parte, cortar um círculo de raio máximo. O círculo formará a base do cilindro.
3. Enrole a segunda parte de tal forma que esta tenha um perímetro de comprimento igual o da circunferência do círculo,e coloque o círculo numa das extremidades do rolo, formando um cilindro. Note que ao enrolar esta segunda parte (o *rolo*) pode-se ter alguma sobreposição das partes afim de obter o comprimento necessário do perímetro (a primeira parte).

Dadas as dimensões da folha de papel, você pode calcular o maior volume possível de um cilindro que pode ser construído utilizando o procedimento descrito acima?

Especificação de Entrada

A entrada consiste de vários casos de teste. Cada caso de teste consiste de dois números `w` e `h` ($1 \leq w \leq h \leq 100$), as quais indicam a largura e altura da folha de papel.

O último caso de é seguido por uma linha contendo dois zeros.

Especificação de Saída

Para cada caso de teste, imprima uma linha com o maior volume possível do cilindro. O arredondamento deste número para 3 casas depois da vírgula.

Exemplo de Entrada	**Exemplo de Saída**

`10 10`
`10 50`
`10 30`
`0 0`
	`54.247`
	`785.398`
	`412.095`

No primeiro caso, o cilindro ótimo tem um raio em torno de 1.591549, no segundo caso, o cilindro ótimo tem um raio de 5, e no terceiro caso, o cilindro ótimo tem um raio de aproximadamente 3,621795.

8 Problema H: Gera Números Aleatórios

John von Neumann sugeriu em 1946 um método para criar uma sequencia de números pseudo-aleatórios. Sua idéia é conhecida como método do “quadrado médio"e funciona do seguinte modo: Escolhemos um valor inicial a_0, que tem uma representação decimal de comprimento no máximo n. Multiplicamos então o valor de a_0 por ele mesmo, adicionamos zeros à esquerda até obtermos uma representação decimal de comprimento $2 \times n$ e tomamos os n dígitos do meio para formar a_i. O processo é repetido para cada a_i com $i > 0$. Neste problema usamos $n = 4$.

Exemplo 1 : $a_0 = 5555, a_0^2 = 30858025, a_1 = 8580, ...$ Exemplo 2: $a_0 = 1111, a_0^2 = 01234321, a1 = 2343, ...$

Infelizmente, este gerador de números aleatórios não é muito bom. Quando partimos com um valor inicial ele não produz outros números com o mesmo número de dígitos. Sua tarefa é checar para um dado valor inicial a_0 quantos números diferentes são produzidos.

Especificação da Entrada

A entrada contém vários casos-teste. Cada caso-teste consiste de uma linha contendo a_0, ($0 < a_0 < 10000$). Os números são possivelmente suplementados com zeros à esquerda tais que cada número consiste de exatamente 4 dígitos. A entrada é terminada com uma linha contendo o valor 0.

Especificação da saída

Para cada caso-teste, imprima uma linha contendo o número de diferentes valores a_i produzidos por este gerador de números aleatórios quando iniciado com o dado valor a_0. Note que a_0 também deve ser contado.

Exemplo de Entrada	**Exemplo de saída**
5555 0815 6239 0	32 17 111

Note que o terceiro caso-teste tem o número máximo de diferentes valores entre todas as possíveis entradas.

9 Problema I: Quem Vai Ser Reprovado?

Prof. Wallywow da Universidade da Columbia Britânica está muito preocupado com a queda do nível de atenção de seus estudantes. Ele já tentou várias técnicas mundialmente conhecidas para incentivar os alunos a prestar atenção nas suas aulas e fazer as tarefas que ele passa para a turma: deu nota para os alunos mais participativos, ofereceu chocolates aos alunos, levou seu karaokê e cantava nas aulas etc. Como tais medidas não levaram a uma melhora no comparecimento às aulas (a idéia do karaokê, inclusive, mostrou-se bastante infeliz... na segunda aula com karaokê a turma reduziu-se a um aluno – que tinha problemas auditivos) ele teve uma brilhante idéia: faria uma competição entre os alunos.

Prof. Wallywow passou um conjunto de problemas aos alunos, e deu um mês para que eles os resolvessem. No final do mês os alunos mandaram o número de problemas resolvidos corretamente. A promessa do brilhante didata era reprovar sumariamente o último colocado da competição. Os alunos seriam ordenados conforme o número de problemas resolvidos, com empates resolvidos de acordo com a ordem alfabética dos nomes (não há homônimos na turma). Isso fez com que alunos com nomes iniciados nas últimas letras do alfabeto se esforçassem muito nas tarefas, e não compartilhassem suas soluções com colegas (especialmente aqueles cujos nomes começassem com letras anteriores). Sua tarefa neste problema é escrever um programa que lê os resultados dos alunos do Prof. Wallywow e imprime o nome do infeliz reprovado.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em um inteiro n ($1 \leq n \leq 100$) indicando o número de alunos na competição. Cada uma das n linhas seguintes contém o nome do aluno e o número de problemas resolvidos por ele. O nome consiste em uma seqüência de letras `[a-z]` com no máximo 20 letras e cada aluno resolve entre 0 à 10 problemas.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o nome do infeliz reprovado.

Após cada instância imprima uma linha em branco.

Exemplo de Entrada	**Exemplo de saída**

`4`
`cardonha 9`
`infelizreprovado 3`
`marcel 9`
`infelizaprovado 3`
	`Instancia 1`
	`infelizreprovado`

10 Problema J: Decorando a Parede

Introdução

Após enriquecer com sua empresa de informática, o Sr. Severino Portas (no nordeste brasileiro, todo Severino é chamado de *Biú*) constuiu a sua casa.

Depois de construir sua enorme vila, Sr. *Biú* Portas percebeu que as paredes interiores estavam muito brancas. Para mudar isso, ele começa a pendurar quadros de sua maravilhosa coleção. Mas logo ele percebe que se torna muito difícil encontrar um lugar na parede, onde uma pintura poderia ser pendurada sem sobrepor outras pinturas.

Agora ele precisa de um programa que lhe diga, dada as pinturadas já colocadas onde colocar as próximas pinturas sem mover quaisquer outras pinturas (ou indicar que isto é impossível).

As pinturas tem forma retangular e devem ser colocadas paralelamente aos lados da parede. O Sr. Portas (em inglês: *gates*) oferece uma bela recompensa a quem ajudá-lo a resolver o problema. Ele lançou esta competição em sua empresa, mas até agora nenhum dos seus melhores programadores tinha resolvido. Será que voce poderia dar uma *mãozinho* para o Sr. *Biú*?

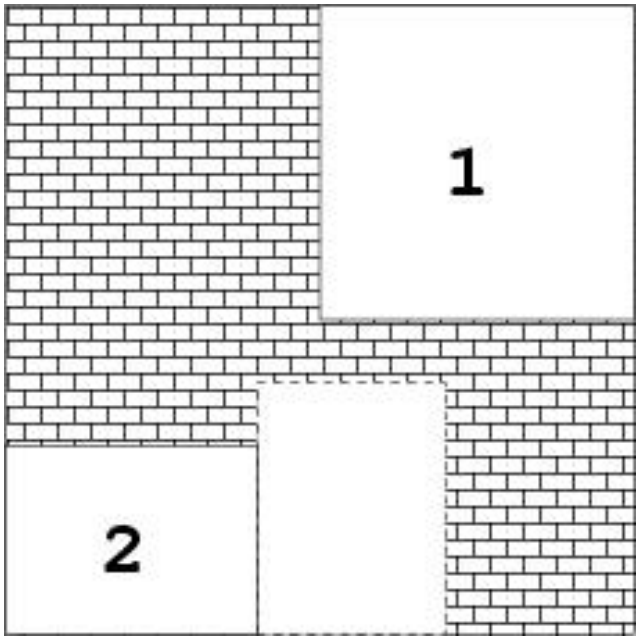

Figura 2: A figura ilustra o segundo caso de teste.

Especificação da Entrada

A primeira linha do arquivo de entrada contém um número que representa o número de casos de testes a serem resolvidos.

Cada caso começa com uma linha contendo três números n, w e h. Onde n é o número de quadros já pendurados na parede, w é a largura da parede e h é a altura da parede.

As próximas n linhas contem 4 inteiros $x1$, $y1$, $x2$, $y2$ onde ($0 \leq x1 < x2 \leq w, 0 \leq y1 < y2 \leq h$), as coordenadas x fornecem a distância da extremidade esquerda da parede, as coordenadas y fornecem a distância para a parte inferior da parede. O par ($x1$, $y1$) corresponde a posição do canto inferior esquerdo de uma pintura, e ($x2$, $y2$) corresponde a posição do canto superior direito.

A última linha de cada caso teste contém as dimensões da próxima pintura a ser pendurada, sendo primeiro informado a sua largura w' e, então a sua altura h' onde ($1 \leq w' \leq w, 1 \leq h' \leq h$).

Você não tem permissão para rotacionar a pintura. Você pode assumir que $0 \leq n \leq 200$ e $1 \leq w, h \leq 1000000$. Além disso, todas as pinturas já penduradas não se sobrepõem.

Especificação da Saída

Produzir uma linha de saída para cada caso de teste. Escrever `Fail!` se não houver lugar disponível na parede em que a pintura poderia ser colocada sem sobreposição de outras pinturas. Caso contrário, escreva as coordenadas onde o canto inferior esquerdo da pintura deve ser colocada. No caso de existir mais de uma solução, escolha a solução com as coordenadas mínima y, e desempate pela coordenada x.

Exemplo de Entrada	Exemplo de Saída
2 1 10 9 5 4 10 9 9 5 2 10 10 5 5 10 10 0 0 4 3 3 4	Fail! 4 0