

Problema A

RSA

Arquivo: rsa.[c|cpp|java]

RSA é um conhecido algoritmo de criptografia de chave pública. Publicado em 1977, ele foi batizado com as iniciais de seus criadores: Ron Rivest, Adi Shamir e Leonard Adleman. Esse tipo de sistema utiliza duas chaves: a pública e a privada. O que é encriptado pela chave pública só pode ser recuperado com a chave privada.

A geração das chaves é feita da seguinte maneira:

- Escolha dois números primos aleatórios p e q ;
- Calcule $n = pq$;
- Compute a função totiente de Euler $\phi(n)$. Como p e q são primos, este valor é dado por $\phi(n) = (p - 1)(q - 1)$;
- Escolha um inteiro e ($1 < e < \phi(n)$) coprimo de $\phi(n)$, ou seja, $\gcd(e, \phi(n)) = 1$;
- Encontre d tal que $(d \cdot e) \bmod \phi(n) = 1$;
- A chave pública é composta por (n, e) e a chave privada por (n, d) .

Roberta trabalha para a Loja de Produtos Realmente Misteriosos (LPRM). Para garantir uma comunicação segura com seus fornecedores ainda mais misteriosos, ela inventou o Roberta's Security Algorithm (RSA) que, curiosamente, funciona exatamente como o RSA, com apenas uma mudança: p e q são sempre números de 32 bits.

Roberta está muito ocupada com seus negócios misteriosos e pediu a sua ajuda para implementar a geração de chaves de seu novo algoritmo.

Entrada

A entrada é composta de três inteiros p , q e e ($1 < p, q, e < 2^{32}$) separados por um único espaço em branco. A entrada termina quando $p = q = e = 0$. Este caso não deve ser processado.

Saída

Para cada caso de teste imprima os dois inteiros que compõem a chave privada (n e d) separados por um único espaço em branco.

Exemplo de entrada

```
5 11 3
1816890443 2715210557 3
0 0 0
```

Saída Para o exemplo de entrada

55 27

4933240111746006751 3288826738142603835

Problema B

Dois Caminhos

Arquivo: doiscaminhos.[c|cpp|java]

O irmão de Bob vive em Flatland. Em Flatland existem N cidades, ligados por $N - 1$ estradas de mão dupla. As cidades são numeradas de 1 a n . Você pode ir de uma cidade para outra em movimento ao longo das estradas.

A empresa Dois Caminhos, onde o irmão de Bob trabalha, ganhou um concurso para reparar dois caminhos em Flatland. Um caminho é uma seqüência de diferentes cidades, ligados sequencialmente por estradas. A empresa tem permissão para escolher por si só os caminhos para reparação. A única condição que têm de satisfazer é que os dois caminhos não devem se cruzar (isto é, não deve ter cidades comuns).

Sabe-se que o lucro que a empresa Dois caminhos terá, é igual ao produto dos comprimentos dos dois caminhos. Vamos considerar o comprimento de cada estrada é igual a 1, e o comprimento de um caminho é igual à quantidade de estradas na mesma. Encontre o máximo lucro possível para a empresa.

Entrada

A entrada é composta por vários testes. A primeira linha de cada teste contém um inteiro n ($2 \leq N \leq 200$), onde N é a quantidade de cidades do país. As seguintes $n - 1$ linhas contêm as informações sobre as estradas. Cada linha contém um par de números das cidades, ligadas por estrada a_i, b_i ($1 \leq a_i, b_i \leq N$). A entrada termina quando $N = 0$.

Saída

Para cada teste, imprimir o máximo de lucro possível para aquele caso.

Exemplo de entrada

```
4
1 2
2 3
3 4
7
1 2
1 3
1 4
1 5
1 6
1 7
6
1 2
2 3
2 4
5 4
6 4
0
```

Saída Para o exemplo de entrada

1
0
4

Problema C

Sapos de Tsé-Tsé

Arquivo: sapos.[c|cpp|java]

A mosca do sono é uma das pragas mais sérias na China, que causa prejuízos enormes ao governo do país. Populações inteiras de pequenas cidades são picadas pela mosca e acabam caindo no sono durante o trabalho (muitos suspeitam que nem são as moscas as causadoras do problema, mas isso é outra história...).

Preocupados com esta situação os pesquisadores de Engenharia Genética da Universidade de Zhao-Zhao estudaram o genoma de um sapo comedor de insetos da região e descobriram que o padrão de saltos do sapo poderia ser facilmente controlado se uma alteração fosse feita em seu cromossomo 12. Infelizmente nem todos os experimentos resultaram em sucesso e, além de alguns sapos sem pernas e com 12 olhos, os experimentos deram origem a várias espécies de sapos com características diferentes de saltos. O objetivo deste problema é que vocês desenvolvam um programa que, a partir da observação do padrão de saltos de um sapo, verifique se ele é do tipo desejado. Um sapo é do tipo desejado se colocado no canto superior esquerdo de um lago retangular ele cobrir toda a extensão do lago com um número mínimo de saltos.

Para anotar o padrão de saltos de um sapo foram feitos vários experimentos. Em cada experimento o sapo foi colocado em uma posição do lago e se anotou para que posição vizinha ele saltou. As posições vizinhas são ordenadas de 1 a 8 no sentido dos ponteiros do relógio, começando na posição imediatamente acima da posição do sapo, como na figura abaixo.

8	1	2
7	sapo	3
6	5	4

Sua tarefa é, dada uma instância de um lago, marcado em cada uma de suas posições com o padrão de saltos do sapo, verificar se este, quando colocado no canto superior esquerdo do lago percorre todas as suas posições.

Entrada

São dadas várias instâncias. Cada instância começa com dois inteiros $0 \leq m \leq 1000$ e $0 \leq n \leq 1000$ que definem a dimensão do lago. Em seguida vêm m linhas com n números inteiros, descrevendo o comportamento do sapo quando colocado naquela posição do lago. Valores $m = n = 0$ indicam o final das instâncias e não devem ser processados.

Saída

Para cada instância solucionada, você deverá imprimir um identificador Instancia h em que h é um número inteiro, seqüencial e crescente a partir de 1. Na linha seguinte, você deve imprimir sim se o sapo passou por todas as m_n posições do lago e nao em caso contrário. Uma linha em branco deve separar a saída de cada instância.

Exemplo de entrada

```
3 3
3 3 5
5 7 7
3 3 3
2 3
4 4 2
1 1 2
0 0
```

Saída Para o exemplo de entrada

```
Instancia 1
sim
```

```
Instancia 2
nao
```

Problema D

Mina de Sal

Arquivo: mineracao.[c|cpp|java]

A cidade de Wieliczka na Polônia é conhecida por sua imponente mina de sal. Reza a lenda que Santa Conegunda, filha de um rei húngaro, foi oferecida em casamento ao rei da Polônia, e seu pai quis oferecer-lhe vários tesouros como dote. A santa recusou, dizendo que preferia receber o seu dote em sal, um bem primordial para a existência humana. O pai lhe deu, então, uma mina de sal na Romênia, onde a santa jogou seu anel de ouro em agradecimento. Anos mais tarde, ao passar pela cidade de Wieliczka, a santa apontou um lugar no solo e pediu que se cavasse ali. Para espanto de todos descobriu-se no local uma mina de sal, e, mais surpreendente, o anel da santa estava ali.

A extração de sal mineral deve respeitar restrições importantes, uma vez que o material deve ser retirado de forma que as paredes e teto da mina permaneçam estáveis e não caiam sobre as cabeças dos mineiros. Estudos de engenheiros de minas da Universidade da Cracóvia mostram que pode-se modelar a produção de sal da mina através de um polinômio $A(x)$ (onde x é o volume total de material retirado da mina). Sabe-se que depois de um período de trabalho, em que foi retirado um volume x_0 de material da mina a produção passa a ser dada por $q(x)$ e existe um desperdício r onde $A(x) = q(x)(x-x_0)+r$.

Sua tarefa neste problema é dados $A(x)$ e um inteiro x_0 , determinar o desperdício r e o novo polinômio relacionado à capacidade de produção da mina $q(x)$.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (EOF). A primeira linha de cada instância contém dois inteiros n e x_0 que representam o grau do polinômio e o valor retirado da mina de sal, respectivamente. A linha seguinte contém $n+1$ inteiros que indicam os coeficientes do polinômio $A(x)$. Isto é, o i -ésimo inteiro da linha seguinte é o coeficiente de x^{i-1} de $A(x)$.

Saída

Para cada instância seu programa deve imprimir duas linhas. A primeira linha deve ser composta de um inteiro indicando o desperdício r . A segunda linha deve conter n inteiros, onde o i -ésimo inteiro é o coeficiente de x^{i-1} de $q(x)$.

Restrições

- $1 \leq n \leq 15$.
- $1 \leq x_0 \leq 50$.
- O coeficiente de cada termo do polinômio $A(x)$ é um inteiro entre 0 e 100.

Exemplo de entrada

```
2 3  
3 2 4  
3 7  
1 10 5 7
```

Saída Para o exemplo de entrada

```
45  
14 4  
2717  
388 54 7
```

Problema E

Sufixo

Arquivo: sufixo.[c|cpp|java]

Dado duas palavras distintas s e t você precisa transformar a palavra s em palavra t . Para isso existem três tipos de operações que podem ser feitas: *automaton*, *array* e *both*. *Automaton* consiste em remover uma letra de s para tentar chegar a t . *Array* consiste em trocar dois caracteres de s para tentar chegar a t . *Both* consiste na combinação de *automaton* e *array*. Sua tarefa consiste em dizer se uma transformação, quando possível, foi somente por *automaton*, somente por *array* ou utilizando as duas. Se não for possível também deve ser informado.

Entrada

A primeira linha com um número inteiro N ($1 \leq N \leq 50$). Em seguida, terão N blocos com 2 linhas cada. A primeira linha de cada bloco conterá uma palavra s não-vazia. A segunda linha de cada bloco conterá uma palavra não vazia t . Palavras s e t são diferentes. Cada palavra consiste apenas em letras minúsculas. Cada palavra contém, no máximo, 100 letras.

Saída

Para cada bloco, imprimir a resposta. Imprimir "need tree" (sem as aspas) se a palavra s não pode ser transformada em palavra t mesmo com o uso de ambos *array* e *automaton*. Imprimir "automaton" (sem as aspas) se você precisa apenas o *automaton* para resolver o problema. Imprimir "array" (sem as aspas) se você precisa apenas o *array* para resolver o problema. Imprimir "both" (sem as aspas), se você precisar de ambas as operações para resolver o problema.

É garantido que, se você pode resolver o problema apenas com o uso de *array*, então é impossível resolvê-lo apenas com o uso do *automaton*. Isto também é verdade para *automaton*.

Exemplo de entrada

```
4
automaton
tomat
array
arary
both
hot
need
tree
```

Saída Para o exemplo de entrada

```
automaton  
array  
both  
need tree
```

Problema F

Estacionamento

Arquivo: estacionamento.[c|cpp|java]

Um estacionamento utiliza um terreno em que os veículos têm que ser guardados em fila única, um atrás do outro. A tarifa tem o valor fixo de R\$10,00 por veículo estacionado, cobrada na entrada, independente de seu porte e tempo de permanência. Como o estacionamento é muito concorrido, nem todos os veículos que chegam ao estacionamento conseguem lugar para estacionar.

Quando um veículo chega ao estacionamento, o atendente primeiro determina se há vaga para esse veículo. Para isso, ele percorre a pé o estacionamento, do início ao fim, procurando um espaço que esteja vago e tenha comprimento maior ou igual ao comprimento do veículo. Para economizar seu tempo e energia, o atendente escolhe o primeiro espaço adequado que encontrar; isto é, o espaço mais próximo do início.

Uma vez encontrada a vaga para o veículo, o atendente volta para a entrada do estacionamento, pega o veículo e o estaciona no começo do espaço encontrado. Se o atendente não encontrar um espaço adequado, o veículo não entra no estacionamento e a tarifa não é cobrada. Depois de estacionado, o veículo não é movido até o momento em que sai do estacionamento.

O dono do estacionamento está preocupado em saber se os atendentes têm cobrado corretamente a tarifa dos veículos estacionados e pediu para você escrever um programa que, dada a lista de chegadas e saídas de veículos no estacionamento, determina o faturamento total esperado.

Entrada

A entrada é composta por diversos casos de teste. A primeira linha de um caso de teste contém dois números inteiros C ($1 \leq C \leq 1000$) e N ($1 \leq N \leq 10000$) que indicam respectivamente o comprimento em metros do estacionamento e o número total de eventos ocorridos (chegadas e saídas) de veículos). Cada uma das N linhas seguintes descreve uma chegada ou saída. Para uma chegada de veículo, a linha contém a letra ' C' ', seguida de dois inteiros P ($1000 \leq P \leq 9999$) e Q ($1 \leq Q \leq 100$), todos separados por um espaço em branco. P indica a placa do veículo e Q o seu comprimento. Para uma saída de veículo, a linha contém a letra ' S' ' seguida de um inteiro P , separados por um espaço em branco, onde P indica a placa do veículo. As ações são dadas na ordem cronológica, ou seja, na ordem em que acontecem.

No início de cada caso de teste o estacionamento está vazio. No arquivo de entrada, um veículo sai do estacionamento somente se está realmente estacionado, e a placa de um veículo que chega ao estacionamento nunca é igual a placa de um veículo já estacionado.

Saída

Para cada caso de teste seu programa deve imprimir uma linha contendo um número inteiro representando o faturamento do estacionamento, em reais.

Exemplo de entrada

```
10 7
C 1234 5
C 1111 4
C 2222 4
C 4321 3
S 1111
C 2002 6
C 4321 3
30 10
C 1000 10
C 1001 10
C 1002 10
S 1000
S 1002
C 1003 20
S 1001
C 1004 20
S 1004
C 1005 30
20 10
C 1234 20
C 5678 1
S 1234
C 1234 20
C 5678 1
S 1234
C 5678 1
C 1234 20
C 5555 1
S 5678
```

Saída Para o exemplo de entrada

```
30
50
40
```

Problema G

Muralha das Ilhas

Arquivo: muralha.[c|cpp|java]

Os bilionários do mundo, donos de pequenos arquipélagos de ilhas, decidiram construir um muro ao redor de suas ilhas para evitar que invasores entrassem em seu território. Portanto eles decidiram calcular quanto seria necessário gastar para proteger cada arquipélago. Para construir o muro aproveitando ao máximo das propriedades de terra, eles decidiram construí-lo na borda de cada ilha. Cada bilionário tem um mapa representado o arquipélago e esse mapa indica se uma determinada posição representa água ou terra. Uma posição do mapa representando terra pode ser definida como borda quando pelo menos uma de suas 8 posições vizinhas representa água. O custo final do muro do arquipélago será dado pelo número de posições no mapa que deverão conter muro futuramente.

Entrada

A entrada consiste de dois inteiros w e h (menores do que 600) definindo respectivamente a largura e a altura do mapa representando o arquipélago. Seguidos de h linhas de w caracteres descrevendo cada posição do mapa. Caracteres com valor 1 representarão água e caracteres com valor 0 representarão terra. Assuma que as posições localizadas nas bordas do mapa sempre conterão agua.

Saída

Sua saída deverá simplesmente informar o custo para proteger o arquipélago.

Exemplo de entrada

```
8 10
11111111
10011111
10001101
10001111
11001111
11111111
11000011
11000001
11000011
11111111
```

```
6 5
111111
100011
100001
100011
111111
```

Saída Para o exemplo de entrada

22

9

Problema H

Contando Subsequências

Arquivo: subsequencias.[c|cpp|java]

Uma subsequência P de uma string Q é uma string que pode ser gerada a partir da remoção de alguns caracteres de Q. Isto é, todos os caracteres de P estão presentes em Q na mesma ordem, porém não necessariamente consecutivos. Por exemplo, "abc" é uma subsequência das strings "zawbycx", "aabbcc" e "zabcz".

Rick e Rose são dois estudantes de Computação que receberam a seguinte tarefa de um professor: contar a quantidade de ocorrências de subsequências em uma lista de strings.

Inicialmente, eles pensaram em fazer a contagem manualmente. Contudo, quando se depararam com strings maiores, Rick e Rose perceberam que precisavam da ajuda de um computador para completarem a tarefa mais rapidamente. Você deve escrever um programa que ajude Rick e Rose, resolvendo a tarefa que o professor passou para eles.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias que seguem. Cada instância é composta por três linhas. A primeira linha de cada instância contém dois inteiros N ($1 \leq N \leq 50$) e M ($1 \leq M \leq N$) representando o tamanho das duas strings a serem testadas. As duas próximas linhas contém duas strings Q e P de tamanho N e M , respectivamente. As strings contém apenas caracteres alfanuméricos.

Saída

Para cada instância, imprima uma linha contendo a quantidade de vezes que P ocorre como subsequência de Q .

Exemplo de entrada

```
5
6 3
aaaaaa
aaa
6 3
abcabc
abc
4 2
aabb
ab
1 1
a
b
1 1
a
a
```

Saída Para o exemplo de entrada

20
4
4
0
1

Problema I

Teclado Maluco

Arquivo: teclado.[c|cpp|java]

Adotou-se um padrão para digitar e guardar as notas de alunos das turmas de universidades. O padrão usa dois pontos, “：“, para definir a separação entre turmas distintas; espaço, “ ”, para separar uma nota da outra, e um ponto, “.”, para indicar o final dos dados. Seu trabalho será fazer um programa para gerar o somatório das notas de cada turma de uma universidade. Por exemplo, para a entrada “ 1 2 2: : 5 1 : 10.”, você deverá gerar a seguinte saída “5:0:6:10”. Perceba, que turmas podem estar vazias. Contudo, um complicador apareceu durante a digitação dos dados de algumas universidades, pois os teclados dos computadores estavam malucos. As vezes o usuário digitava espaço “ ”, e o teclado disparava uma sequência de caracteres juntamente ou no lugar do espaço digitado. Portanto seu programa deverá considerar essa situação.

Entrada

A entrada consiste de um texto contendo sequências de números separados por dois pontos “：“ e terminado por um ponto “.”. Os números são separados por um espaço “ ”, ou qualquer outra sequência de caracteres (não excedendo 500 caracteres), exceto caracteres numéricos, dois pontos, ou um ponto. Considere que as notas variam de 0 a 10.

Saída

Sua saída deverá informar a soma das notas de cada turma separadas por dois pontos “：“.

Exemplo de entrada

```
10qwef2 wqe
7 o:iq
hbfqwe
qfcwqe
1qwefqwe2m; ,jfg
:3fhbr
fdsbhr [
2rthr
1.

qwef2 wqefwqe3plknger
1 *-o:i::q
4hbfqwe
qfcwqe
1qwefqwe10m; ,jfg
:2 3:3fhbr
fdsbhr [
2rthr
1.
```

Saída Para o exemplo de entrada

19:3:6

6:0:0:15:5:6

Problema J

As Fazendas

Arquivo: fazendas.[c|cpp|java]

Um problema comum para alguns fazendeiros é a marcação dos limites de seus terrenos.

Uma prática comum é ver fazendas com seus terrenos delimitados por um conjunto de estacas com arames farpados ligando elas.

Observando essa marcação de terreno em uma visão 2D, um mapa, observamos que é formado um polígono.

Seu objetivo aqui é ajudar dois fazendeiros a prever se suas fazendas estão com alguma região em comum, sendo que ambos receberam os dados de localização das estacas da prefeitura.

Veja dois exemplos ilustrando possíveis situações:

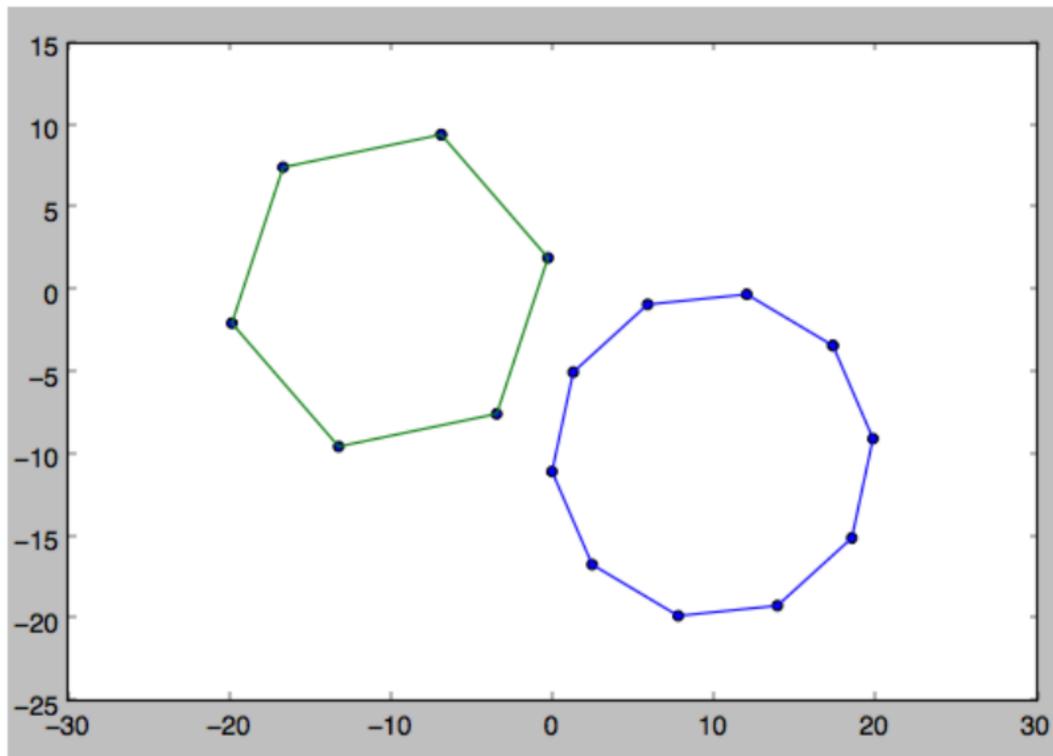


Figure 1: Fazendas sem região em comum

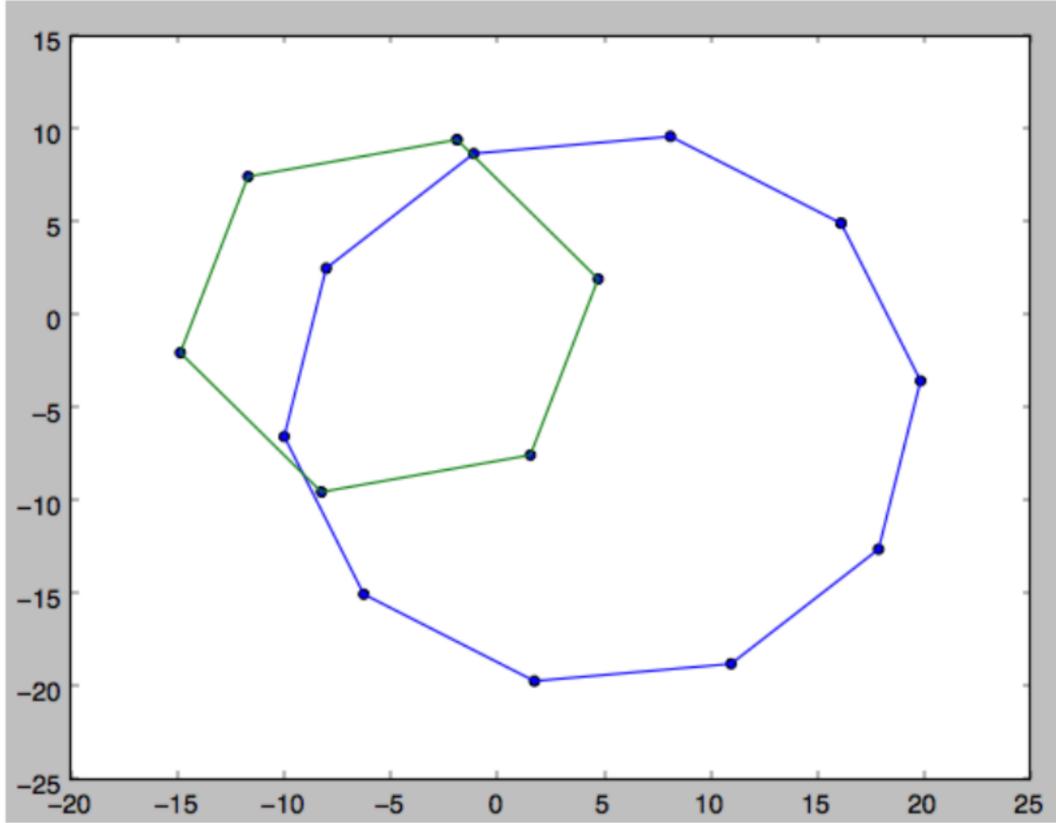


Figure 2: Fazendas com região em comum formando polígono com 7 pontos

Entrada

A primeira linha de cada caso de teste possui dois números inteiros N e M .

Onde N e M indicam a quantidade de estacas da fazenda A e B, respectivamente.

As $N+M$ linhas seguintes são as coordenadas cartesianas (x, y) das estacas.

Seu programa deve encerrar a execução quando N e M forem iguais a zero.

$$3 \leq N, M \leq 1000.$$

x e y são pontos flutuantes.

Saída

A cada linha deve ser apresentado o numero de pontos do polígono que representa a região comum entre as fazendas para cada caso de teste.

Considerações gerais

- Precisão de 1.10^{-10}
- Use o tipo "double" para os pontos flutuantes
- O sistema é euclidiano
- O polígono gerado pela cerca sempre será convexo

Exemplo de entrada

```
4 4
0 0
0 1
1 1
1 0
-0.5 -0.5
-0.5 0.5
0.5 0.5
0.5 -0.5
0 0
```

Saída Para o exemplo de entrada

```
4
```