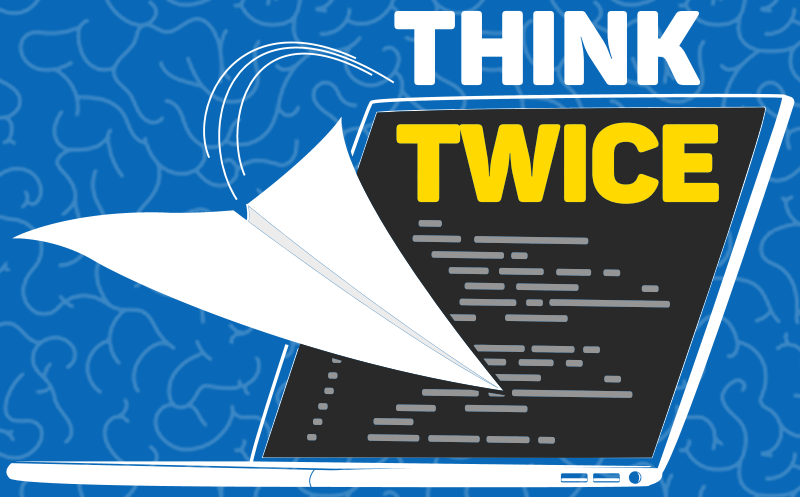
ENUNCIADO

EDIÇÃO 2020



GUIDELINES

De forma a agilizar o processo de correção dos exercícios a organização decidiu desenvolver uma plataforma de correção automática. Por este motivo existem certas guidelines que devem ser cumpridas de forma a que os exercícios sejam corrigidos corretamente:

- O ficheiro a ser executado deve ter o nome “main”;
- O input deve ser sempre lido a partir de um ficheiro .txt e o nome desse ficheiro deve ser passado como argumento ao ficheiro executavel.
- O output deve ser sempre escrito para um ficheiro com o nome “result.txt” e o seguinte path deve ser passado à função que irá escrever o output. Path: “teamX/challengeY/result.txt” em que X é o restante nome do repositório da equipa e Y o número do desafio.
- Para os desafios resolvidos em C ou C++ é necessário existir um ficheiro main.c ou main.cpp e um makefile que irá ser executado e irá gerar um ficheiro com o nome “main” para ser executado pela plataforma.
- Não alterar o nome do repositório nem o nome das várias pastas dos exercícios. Se pretenderem alterar algo no repositório falem primeiro com a organização.

Devido à estrutura dos exercícios 25 e 36, estes exercícios terão que ser corrigidos manualmente. Por isso a resolução destes exercícios terá que ser submetida até às 9h00 do dia 8 de março, domingo para ser corrigida.

Estamos a usar pela primeira vez uma plataforma de correção automática neste evento e por isso poderão existir falhas na mesma. Por este motivo, pedimos que falem com um membro da organização quando verificarem inconsistências na correção dos exercícios para que essas situações possam ser analisadas.

CREDENCIAIS DE ACESSO INTERNET

USER: event2@ua.pt

PASSWORD: ttw#2020

DIFICULDADE

BAIXA

EXERCÍCIO 1

DESCRIÇÃO:

O trabalho na área da criptografia resultou no uso prático de resultados da teoria dos números e de outros ramos da matemática que antes eram considerados apenas de interesse teórico.
Este problema envolve a computação eficiente de raízes inteiras de números. Dado um número inteiro $n \geq 1$ e um número inteiro $p \geq 1$, escreva um programa que determine $\sqrt[n]{p}$, a n-ésima raiz positiva de p. Neste problema, dados n e p, p é sempre do tipo k^n para um inteiro k (este inteiro é o que o teu programa deve encontrar).

INPUT:

O input consiste numa sequência de pares inteiros n e p com cada número inteiro numa linha por si só. Para todos esses pares $1 \leq n \leq 200$, $1 \leq p < 10101$ e existe um número inteiro k, $1 \leq k \leq 109$ tal que $k^n = p$.

OUTPUT:

Para cada par inteiro n e p, o valor np deve ser impresso, ou seja, o número k tal que $k^n = p$.

EXEMPLO DE INPUT 1:

2
16

EXEMPLO DE INPUT 2:

3
27

EXEMPLO DE OUTPUT 1:

4

EXEMPLO DE OUTPUT 2:

3

EXERCÍCIO 2

DESCRIÇÃO:

TEX é uma linguagem tipográfica desenvolvida por Donald Knuth. Recebe texto fonte com algumas instruções e produz um belo documento. Documentos bonitos usam “ e ” para delimitar citações, em vez do normal ", que é fornecido pela maioria dos teclados.
Os teclados normalmente não possuem aspas duplas orientadas, mas possuem aspas simples ` e aspas simples à direita '.

Deves escrever um programa que converta texto que contém caracteres de aspas duplas (") em texto que seja idêntico, exceto que as aspas duplas foram substituídas. Os caracteres de aspas duplas (") devem ser substituídos adequadamente por `` se o " abrir uma cotação e por '' (duas single quotes) se o " fechar uma cotação.

INPUT:

A entrada consistirá em texto que contém um número par de caracteres de aspas duplas (").

OUTPUT:

O texto deve ser impresso exatamente como foi inserido, exceto pelo seguinte:

- o primeiro " em cada par é substituído por dois caracteres `
- o segundo " em cada par é substituído por dois caracteres '

EXEMPLO DE INPUT 1:

"To be or not to be," quoth the Bard, "that is the question".

EXEMPLO DE INPUT 2:

The programming contestant replied: "I must disagree. To `C' or not to `C', that is The Question!"

EXEMPLO DE OUTPUT 1:

``To be or not to be,'' quoth the Bard, ``that is the question''.

EXEMPLO DE OUTPUT 2:

The programming contestant replied: ``I must disagree. To `C' or not to `C', that is The Question!''

EXERCÍCIO 3

DESCRIÇÃO:

Numa antiga estação ferroviária, ainda podemos encontrar um dos últimos swappers de comboios restantes. Um swapper de comboios é um serviço da ferrovia, cujo único trabalho é reorganizar as carruagens dos comboios.
Uma vez que as carruagens estão organizadas na ordem ideal, tudo o que o maquinista deve fazer é largar as carruagens, uma a uma, nas estações para as quais a carga é destinada.
O título swapper de comboios deriva da primeira pessoa que executou essa tarefa, numa estação próxima a uma ponte ferroviária. Em vez de abrir verticalmente, a ponte girava em torno de um pilar no centro do Rio. Depois de girar a ponte 90 graus, os barcos podiam passar para a esquerda ou direita.
O primeiro "swapper de comboios" descobriu que a ponte podia ser operada com no máximo duas carruagens nela. Ao girar a ponte 180 graus, as carruagens mudavam de lugar, permitindo assim que ele reorganizasse as carruagens (como efeito colateral, as carruagens ficavam na direção oposta, mas as carruagens de comboios podem mover-se em qualquer sentido, então não era importante).
Agora que quase todos os "swappers de comboios" morreram, a empresa ferroviária gostaria de automatizar a sua operação. Parte do programa a ser desenvolvido, é uma rotina que decide, para um determinado comboio o número mínimo de trocas (swaps) de duas carruagens adjacentes para ordenar o comboio. A tua tarefa é criar essa rotina.

INPUT:

Cada caso de teste consiste na entrada de duas linhas. A primeira linha de um caso de teste contém um número inteiro L, que representa o comprimento do comboio ($0 \leq L \leq 50$).
A segunda linha de um caso de teste contém uma permutação dos números 1 a L, indicando a atual ordem das carruagens. As carruagens devem ser ordenadas de modo a que a carruagem 1 seja o primeiro, depois o 2 etc. com a carruagem L a chegar por último.

OUTPUT:

Para cada caso de teste, imprima a frase: "Optimal train swapping takes S swaps.", onde S é um inteiro.

EXEMPLO DE INPUT 1:

3
1 3 2

EXEMPLO DE INPUT 2:

4
4 3 2 1

EXEMPLO DE OUTPUT 1:

Optimal train swapping takes 1 swaps.

EXEMPLO DE OUTPUT 2:

Optimal train swapping takes 6 swaps.

EXERCÍCIO 4

DESCRIÇÃO:

Neste problema escreva um programa que "cortará" um número de números primos de uma lista de números primos entre (e inclusive) 1 e *N*. O teu programa lerá um número *N*; determina a lista de números primos entre 1 e *N* e imprime os 2*C* números primos do centro da lista caso haja um número par de números primos ou 2*C* - 1 números primos do centro da lista caso haja um número ímpar de números primos na lista.

INPUT:

Cada entrada estará numa linha por si só e será composta por 2 números. O primeiro número ($1 \leq N \leq 1000$) é o número máximo na lista completa de números primos entre 1 e *N*. O segundo número ($1 \leq C \leq N$) define os 2*C* números primos a serem impressos a partir do centro da lista caso o comprimento do lista for par, ou os 2*C* - 1 números a serem impressos a partir do centro da lista, caso o comprimento da lista seja ímpar.

OUTPUT:

Para cada input, deves imprimir o número *N* começando na coluna 1, seguido de um espaço, depois pelo número *C* e depois por dois pontos (:) e depois pelos números centrais da lista de números primos como definido acima. Se o tamanho da lista central exceder os limites da lista de números primos entre 1 e *N*, a lista de números primos entre 1 e *N* (inclusive) deve ser impressa. Cada número do centro da lista deve ser precedido por exatamente um espaço em branco.

EXEMPLO DE INPUT 1:

21 2

EXEMPLO DE INPUT 2:

18 2

EXEMPLO DE OUTPUT 1:

21 2: 5 7 11

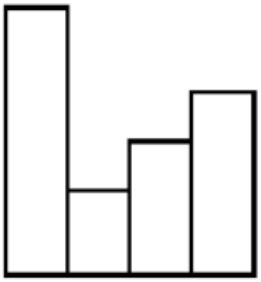
EXEMPLO DE OUTPUT 2:

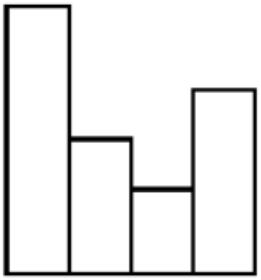
18 2: 3 5 7 11

EXERCÍCIO 5

DESCRIÇÃO:

Existem postes de altura 1, 2,. . . , *n* numa linha. Se olhares para estes postes do lado esquerdo ou direito, postes menores estão ocultos por postes mais altos. Por exemplo, considera os dois arranjos de 4 postes na próxima figura:





Para cada arranjo, apenas um poste pode ser visto da esquerda e dois postes do lado esquerdo.

Escreve um programa para calcular o número de arranjos de *n* postes, tais que visto da esquerda se observem *I* postes e visto da direita se observem *r* postes.

INPUT:

O programa é para ler a partir dos ficheiros de input fornecidos. Cada caso de teste consiste numa linha que contém três números inteiros, *n*, *l* e *r* (1 ≤ *l*, *r* ≤ *n* ≤ 20), em que *n* é o número de postes, *l* é o número de postes vistos da esquerda e r é o número de postes vistos da direita.

OUTPUT:

Imprime exatamente uma linha para cada caso de teste. A linha deve conter o número de arranjos de postes para o caso de teste.

EXEMPLO DE INPUT 1:

4 1 2

EXEMPLO DE INPUT 2:

4 1 1

EXEMPLO DE OUTPUT 1:

2

EXEMPLO DE OUTPUT 2:

0

EXERCÍCIO 6

DESCRIÇÃO:

Nos bons velhos tempos, quando as crianças suecas explodiam foguetes, grupos de crianças excitadas atormentavam algumas cidades durante a Páscoa, com apenas uma coisa em mente: Explodir coisas. As caixas pequenas eram fáceis de explodir e, portanto, as caixas de correio tornaram-se num alvo popular.

Agora, um pequeno fabricante de caixas de correio está interessado em quantos foguetes a sua nova caixa de correio (protótipo) pode suportar sem explodir e contratou-te para ajudá-lo. Ele irá fornecer-te k ($1 \leq k \leq 10$) protótipos de caixa de correio idênticos, cada um com capacidade para até m ($1 \leq m \leq 100$) foguetes. No entanto, ele não tem a certeza de quantos foguetes precisa de fornecer para que possas resolver o problema dele, então ele pergunta. Pensas um pouco e depois dizes: "Bem, se eu explodir uma caixa de correio, não posso usá-la novamente; portanto, se me forneceres apenas caixas de correio $k = 1$, eu teria que começar a testar 1 foguete, depois 2 foguetes, etc. até finalmente explodir. Na pior das hipóteses, isto é, se não explodir mesmo quando cheio de m foguetes, eu precisaria de $1 + 2 + 3 + \ldots + m = m * (m + 1) / 2$ foguetes.

Se $m = 100$, teríamos mais de 5000 foguetes! ". "Isso é demais", ele responde. "E se
eu lhe der mais do que $k = 1$ caixas de correio? Consegues encontrar uma estratégia que exija menos foguetes?"

Consegues? E qual é o número mínimo de foguetes que deves pedir que ele forneça?

Podes assumir o seguinte:

1. Se uma caixa de correio pode suportar x foguetes, também pode suportar x - 1 foguetes.
2. Após uma explosão, uma caixa de correio é totalmente destruída ou ilesa, o que significa que pode ser reutilizada noutra explosão de teste.

Nota: Se a caixa de correio suportar uma carga completa de m foguetes, então o fabricante obviamente fica satisfeito com essa resposta. Mas, caso contrário, ele está à procura do número máximo de foguetes que as suas caixas de correio podem suportar.

INPUT:

Cada caso de teste é descrito por uma linha contendo dois números inteiros: k e m, separados por um único espaço.

OUTPUT:

Para cada caso de teste, imprima uma linha com um único número inteiro indicando o número mínimo de foguetes necessários, na pior das hipóteses, para descobrir quantos foguetes o protótipo da caixa de correio pode suportar.

EXEMPLO DE INPUT 1:

1 10

EXEMPLO DE INPUT 2:

1 100

EXEMPLO DE OUTPUT 1:

55

EXEMPLO DE OUTPUT 2:

5050

EXERCÍCIO 7

AUTOR: Professor Mário Antunes

DESCRIÇÃO:

Dada uma lista desordenada calcular o valor da mediana sem ordenar os valores. O algoritmo deverá ter uma complexidade melhor que O(n).

Considerações: a linha de input serão os números que compõem a lista. O output será a mediana da lista.

EXEMPLO DE INPUT 1:

6 1 3

EXEMPLO DE INPUT 2:

6 1 3 4

EXEMPLO DE OUTPUT 1:

3

EXEMPLO DE OUTPUT 2:

3.5

EXERCÍCIO 8

AUTOR: Professor Mário Antunes

DESCRIÇÃO:

Num tabuleiro de xadrez (oito por oito) navegar em todas as células com um cavalo, sem repetir nenhuma. Considere que cada célula é indexada por um único valor.

Considerações: a linha de input serão as coordenadas iniciais do cavalo. o output será a sequência de posições percorrer o tabuleiro.
Link: http://www.maths-resources.com/knights/

EXEMPLO DE INPUT 1:

0

EXEMPLO DE INPUT 2:

31

EXEMPLO DE OUTPUT 1:

00 15 02 21 18 25 12 23
03 20 17 28 13 22 31 26
16 01 14 19 30 27 24 11
45 04 29 52 47 56 63 32
60 51 46 55 62 53 10 39
05 44 61 48 57 38 33 36
50 59 42 07 54 35 40 09
43 06 49 58 41 08 37 34

EXEMPLO DE OUTPUT 2:

26 23 12 41 02 05 10 07
13 42 25 22 11 08 01 04
24 27 40 45 52 03 06 09
43 14 63 50 21 46 53 00
28 39 44 57 62 51 20 47
15 34 31 38 49 56 59 54
32 29 36 17 58 61 48 19
35 16 33 30 37 18 55 60

EXERCÍCIO 9

DESCRIÇÃO:

Os números binários e o seus padrões de bits são sempre muito interessantes para os programadores de computador. Neste problema, precisas de contar o número de números binários positivos que possuem as seguintes propriedades:

- Os números têm exatamente N bits e não possuem zeros à esquerda.
- A frequência de zeros e uns é igual.
- Os números são múltiplos de K

Figura: Aqui está uma tabela com os números possíveis para alguns dos casos de teste:

6 3	6 4	6 2
101010	111000	111000
	110100	110100
	101100	101100
		110010
		101010
		100110

INPUT:

A entrada para cada caso de teste consiste em dois números inteiros, N ($1 \leq N \leq 64$) e K ($0 \leq K \leq 100$).

OUTPUT:

Para cada input imprima o número de números binários que tem a propriedade que foi mencionada.

EXEMPLO DE INPUT 1:

6 3

EXEMPLO DE INPUT 2:

6 4

EXEMPLO DE OUTPUT 1:

1

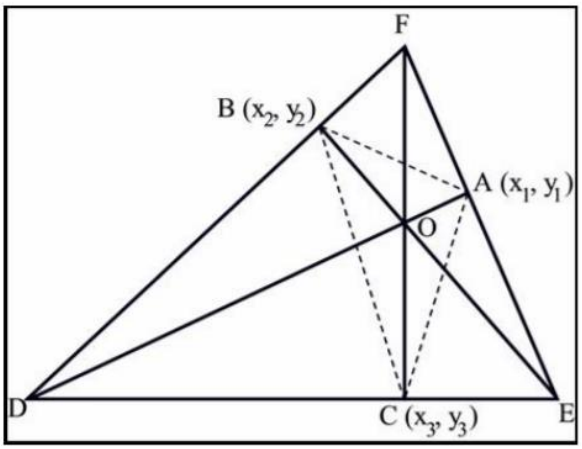
EXEMPLO DE OUTPUT 2:

3

EXERCÍCIO 10

DESCRIÇÃO:

Se DEF é um triângulo agudo e DA, EB e FC são as suas três alturas em EF, DF e DE, respectivamente, então o triângulo ABC é chamado triângulo de altitude do triângulo DEF. É sabido que DA, EB e FC são concorrentes e vamos assumir que o seu ponto comum de interseção é O. Portanto, o ponto O é chamado de ortocentro do triângulo DEF. Pode-se provar que O é o centro do triângulo ABC. Neste problema, receberás o triângulo ABC e o teu trabalho é descobrir o correspondente triângulo agudo DEF.



INPUT:

Cada linha contém seis números inteiros *x1*, *y1*, *x2*, *y2* e *x3*, *y3*. Esses seis números inteiros indicam um triângulo de altitude com os vértices A (*x1*, *y1*), B (*x2*, *y2*) e C (*x3*, *y3*), respectivamente.

OUTPUT:

Para cada linha de input, produza três linhas de output. A descrição dessas três linhas é dada abaixo:

Cada uma das três linhas contém dois números de vírgula flutuante, que são na verdade as coordenadas de D, E e F, respectivamente. Observe que para um determinado triângulo ABC, pode haver quatro possíveis triângulos DEF. Mas és solicitado apenas a encontrar o que é agudo. Erros de precisão não deverão ocorrer se usares números de vírgula flutuante de precisão dupla. Os valores absolutos de nenhum dos números de output serão maiores 100000 e todos os números devem ter três dígitos após a vírgula decimal.

EXEMPLO DE INPUT 1:

682 1369 3981 1233 4333 4583

EXEMPLO DE INPUT 2:

4131 734 1249 4705 2815 475

EXEMPLO DE OUTPUT 1:

6539.582 3443.107
-1528.155 7610.801
1491.578 -917.367

EXEMPLO DE OUTPUT 2:

-1810.802 3068.269
3810.093 -82.858
6872.845 7713.274

EXERCÍCIO 11

DESCRIÇÃO:

O número 729 pode ser escrito como uma potência de várias maneiras: 3^6 9^3 e 27^2. Pode ser escrito como 729^1, é claro, mas isso não conta como uma potência. Queremos dar mais alguns passos. Para isso, é conveniente usar '^' para exponenciação, então definimos a ^ b = a^b. O número 256 também pode ser escrito como 2 ^ 2 ^ 3 ou como 4 ^ 2 ^ 2. Lembra-te de que '^' é associativo à direita, então 2 ^ 2 ^ 3 significa 2 ^ (2 ^ 3).

Definimos uma torre de potências de altura *k* como uma expressão da forma *a1* ^ *a2* ^ *a3* ^. . . ^ a_k, com *k*> 1 e números inteiros a_i > 1.

Dada uma torre de potências de altura 3, representando algum número inteiro n, quantas torres de potências de altura pelo menos 3 representa *n*?

INPUT:

O input contém vários casos de teste, cada um numa linha separada. Cada caso de teste tem o formato *a* ^ *b* ^ *c*, onde *a*, *b* e *c* são inteiros, 1 <*a*, *b*, *c* ≤ 9585.

OUTPUT:

Para cada caso de teste, imprima o número de maneiras pelas quais o número *n* = *a* ^ *b* ^ *c* pode ser representado como uma torre de potências de altura pelo menos três.
O número mágico 9585 é cuidadosamente escolhido de forma que a saída seja sempre menor que 2^{63}.

EXEMPLO DE INPUT 1:

4^2^2

EXEMPLO DE INPUT 2:

8^12^2

EXEMPLO DE OUTPUT 1:

2

EXEMPLO DE OUTPUT 2:

10

EXERCÍCIO 12

DESCRIÇÃO:

Um problema clássico da teoria dos números é "Encontre o número de zeros à direita em Nₘ, na base B". Este problema é bastante convencional e fácil. Mas um número pode ter o mesmo número de zeros à direita em mais do que uma base. Por exemplo, se o número decimal 24 for escrito em base 3, 4, 6, 8, 12 e 24, parecerá 80, 60, 40, 30, 20 e 10, respectivamente. Portanto, em todos os sistemas numéricos, há apenas um zero à direita. Dado um número Nₘ, o teu trabalho é descobrir o número de bases inteiras nas quais ele tem exatamente T zeros à direita.

INPUT:

Cada linha contém três números inteiros N ($1 \le N \le 10^8$), M ($0 < M \le 10^7$) e T ($0 < T \le 10^4$). O significado de N, M e T é dado na declaração do problema.

OUTPUT:

Para cada linha de input, produza uma linha de output. Esta linha contém um número inteiro N B, que é o módulo 100000007 do valor do número de bases em que Nₘ tem exatamente T zeros à direita.

EXEMPLO DE INPUT 1:

24 1 1

EXEMPLO DE INPUT 2:

100 200 10

EXEMPLO DE OUTPUT 1:

6

EXEMPLO DE OUTPUT 2:

312

EXERCÍCIO 13

AUTOR: Professor Tomás Oliveira e Silva

DESCRIÇÃO:

Um grande terreno agrícola de forma retangular está dividido em pequenas parcelas quadradas. Sabe-se quanto é que cada uma dessas parcelas rende (pode dar lucro ou prejuízo). Um lavrador pretende comprar parte desse terreno agrícola. Por imposições legais, o terreno que vai ser comprado também tem de ter uma forma retangular, e cada parcela quadrada tem de ser comprada integralmente.

Considerando que não existem problemas de acessibilidade a qualquer zona do terreno, que parte do terreno deve comprar o agricultor de modo a que a zona escolhida tenha o maior lucro possível?

Exemplo (com 3x5 parcelas quadradas, os casos de teste a sério têm até no máximo 100x100 parcelas quadradas):

- número de linhas: 3
- número de colunas: 5
- mapa de lucros

-10	4	38	22	3
-1	27	-10	24	-5
-30	-4	-15	-8	7

- melhor lucro: 105
- zona a comprar: começa na linha 0 e coluna 1 e acaba na linha 1 e coluna 3. (Os números das linhas e colunas começam em 0.)

-10	4	38	22	3
-1	27	-10	24	-5
-30	-4	-15	-8	7

Notas:

- O grande terreno pode ter até 100x100 pequenas parcelas quadradas.
- O lucro de cada parcela, um número inteiro, não pode ser inferior a -100 nem superior a +100.
(Se for negativo, a parcela dá prejuízo.)
- para cada um dos casos de teste a solução é única.
- Os dados do problema são lidos do standard input.
- O programa tem de enviar para o standard output duas linhas de texto com
– na primeira linha, o lucro da zona comprada, e

– na segunda linha, as coordenadas l0, c0, l1 e c1 da zona comprada. As coordenadas começam em 0.

EXEMPLO DE INPUT:

3 5
-10 4 38 22 3
-1 27 -10 24 -5
-30 -4 -15 -8 7

EXEMPLO DE OUTPUT:

105
0 1 1 3

DIFICULDADE MÉDIA

EXERCÍCIO 14

AUTOR: **Professor** Mário Antunes

DESCRIÇÃO:

Dado um conjunto de regiões adjacentes e um conjunto de cores deverá ser atribuída uma cor a cada região sem repetir cores em regiões adjacentes.

Considerações: a primeira linha do input define as cores a ser usadas. As seguintes linhas indicam as adjacências das regiões, uma adjacência por linha. O output a cor de cada região.

EXEMPLO DE INPUT 1:

```
red green blue
WesternAustralia NorthernTerritory
WesternAustralia SouthAustralia
SouthAustralia NorthernTerritory
Queensland NorthernTerritory
Queensland SouthAustralia
Queensland NewSouthWales
NewSouthWales SouthAustralia
Victoria SouthAustralia
Victoria NewSouthWales
Victoria Tasmania
```

EXEMPLO DE INPUT 2:

```
red
Western Australia Northern Territory
Western Australia South Australia
South Australia Northern Territory
Queensland Northern Territory
Queensland South Australia
Queensland New South Wales
New South Wales South Australia
Victoria South Australia
Victoria New South Wales
Victoria Tasmania
```

EXEMPLO DE OUTPUT 1:

```
WesternAustralia red
NorthernTerritory green
```

SouthAustralia blue
Queensland red
NewSouthWales green
Victoria red
Tasmania green

EXEMPLO DE OUTPUT 2:

EXERCÍCIO 15

DESCRIÇÃO:

O objetivo é chegar de um inteiro a outro através de um step que corresponde a um inteiro. O comprimento de um step deve ser não-negativo e pode ser maior que, igual a ou menor que o comprimento da etapa anterior por 1. Qual é o número mínimo de etapas para passar de *x* para *y*? O comprimento do primeiro e do último passo deve ser 1.

INPUT:

Para cada caso de teste, uma linha apresenta dois números inteiros: $0 \leq x \leq y < 2^{31}$.

OUTPUT:

Para cada caso de teste, imprima uma linha fornecendo o número mínimo de etapas para ir de *x* até *y*

EXEMPLO DE INPUT 1:

45 48

EXEMPLO DE INPUT 2:

45 49

EXEMPLO DE OUTPUT 1:

3

EXEMPLO DE OUTPUT 2:

3

EXERCÍCIO 16

DESCRIÇÃO:

Suhan e Laina vivem numa cidade n-dimensional, onde há n + 1 locais. Quaisquer dois locais (considere esses locais como pontos) são equidistantes entre si e conectados por apenas uma estrada bidirecional. Elas adoram andar juntas pela cidade com as suas trotinetes. Kiri, um robô da décima geração, também vive na mesma cidade e quer matar Suhan por ciúmes. É por isso que Suhan e Laina são muito cuidadosas em manter os seus pensamentos e planos em segredo.
Portanto, ninguém sabe:

a) Onde Suhan e Laina vivem.
b) Qual é o local de destino delas.
c) Que estradas é que elas usam?

Assim, a jornada pode começar em qualquer local, terminar noutro local e elas podem usar qualquer sequência de estradas que gostarem. O local de destino pode ser igual ou diferente do local de origem. Por exemplo, quando a rota é garantida como sendo um ciclo simples, a origem e local de destino são os mesmos.

Dado o número de locais na cidade (*L*), terás que encontrar o custo esperado (geralmente considerado em média) de uma das suas viagens individuais. Podes supor que o custo de viajar de um local para outro pelo caminho direto (também o mais curto) é 1 joule universal.

INPUT:

O input contém várias linhas de entrada. Cada linha contém um único número inteiro *L*
(15 ≥ *L*> 2) que indica o número de locais na cidade.

OUTPUT:

O output contém três números de vírgula flutuante *F1*, *F2*, *F3*. Aqui *F1* é o custo esperado quando elas viajam ao longo de um caminho, *F2* é o custo esperado quando é garantido que eles viajam por um **caminho** simples e *F3* é o custo esperado quando é garantido que elas viajam ao longo de um **ciclo** simples. Todos os números de vírgula flutuante devem ser arredondados para quatro dígitos após a vírgula decimal. Deves assumir que o custo da viagem não é maior que *L*. O custo de viagem é sempre expresso em joule universal.

EXEMPLO DE INPUT 1:

3

EXEMPLO DE INPUT 2:

4

EXEMPLO DE OUTPUT 1:

2.4286 1.5000 3.0000

EXEMPLO DE OUTPUT 2:

3.5500 2.2000 3.5000

EXERCÍCIO 17

DESCRIÇÃO:

A “Really Neato Calculator Company, Inc.” contratou recentemente a tua equipa para ajudar a projetar a sua calculadora “Super Neato Model I”. Como cientista da computação, tu sugeriste à empresa que seria interessante que essa nova calculadora pudesse converter entre bases numéricas. A empresa achou que era uma ideia excelente e solicitou à tua equipa que apresentasse o programa de protótipo que realiza a conversão de base. O gerente de projeto da calculadora “Super Neato Modelo I” informou que a calculadora terá as seguintes funcionalidades:

- Terá um display de 7 dígitos.
- Os seus botões incluirão as letras maiúsculas A a F, e dígitos de 0 a 9.
- Terá suporte para bases 2 a 16

INPUT:

O teu programa de protótipo consistirá numa conversão de base por linha. Haverá três números por linha. O primeiro número será o número na base da qual tu estás a converter. Pode ter "0"s iniciais. O segundo número é a base da qual estás a converter. O terceiro número é a base para a qual irás converter. Os números são separados por um espaço em branco.

OUTPUT:

O output será apenas o número convertido, como apareceria no visor da calculadora. O número deve estar justificado à direita no visor de 7 dígitos. Se o número for grande demais para aparecer no visor, imprima 'ERROR' (sem as aspas) justificado à direita no visor.

EXEMPLO DE INPUT 1:

1111000 2 10

EXEMPLO DE INPUT 2:

2102101 3 15

EXEMPLO DE OUTPUT 1:

120

EXEMPLO DE OUTPUT 2:

7CA

EXERCÍCIO 18

AUTOR: Professor Mário Antunes

DESCRIÇÃO:

Num tabuleiro com dimensões configuráveis navegar em todas as células com o movimento em L típico do cavalo, sem repetir nenhuma. Considere que cada célula é indexada por um único valor.

CONSIDERAÇÕES: Alinha de input será as dimensões do tabuleiro e as coordenadas iniciais do cavalo. o output será a sequência de posições percorrer o tabuleiro.

EXEMPLO DE INPUT 1
5 5 0

EXEMPLO DE INPUT 2
6 6 31

EXEMPLO DE OUTPUT 1
00 19 14 09 06
13 24 07 20 15
18 01 10 05 08
23 12 03 16 21
02 17 22 11 04

EXEMPLO DE OUTPUT 2
28 11 14 21 26 31
13 22 27 30 15 20
10 29 12 23 32 25
01 04 07 34 19 16
06 09 02 17 24 33
03 00 05 08 35 18

EXERCÍCIO 19

AUTOR: Professor Mário Antunes

DESCRIÇÃO:

Dado uma configuração inicial das torres de hanoi, escrever um algoritmo de pesquisa que seja capaz de fornecer os passos necessário para resolver as torres.

CONSIDERAÇÕES:

Cada linha de input representa um pino do jogo das torres de hanói, os valores em cada linha representa as peças. Quando maior o valor, maior será a peça. Para terminar a inserção de dados use um terminador "-". O output será de passos necessário para mover todas as peças para o pino mais a direita.

EXEMPLO DE INPUT 1:

3 2 1

-

EXEMPLO DE INPUT 2:

3
2
1
-

EXEMPLO DE OUTPUT 1:

1 -> 2
2 -> 1
1 -> 1
3 -> 2
1 -> 0
2 -> 2
1 -> 2

EXEMPLO DE OUTPUT 2:

3

EXERCÍCIO 20

AUTOR: Professor Carlos Costa

DESCRIÇÃO:

Dado um certo labirinto, o objetivo é obter o caminho mais curto entre “S” (inicio) e “X” (fim).

SUGESTÃO: Utilizar backtracking.

INPUT:

No repositório.

OUTPUT:

No repositório.

EXERCÍCIO 21

AUTOR: Professor Carlos Costa

DESCRIÇÃO:

Dado um ficheiro de texto, o objetivo é obter as palavras que procedem uma determinada palavra.

NOTA: Só se consideram as palavras com mais de três carateres.

INPUT :

No repositório.

OUTPUT:

No repositório.

EXEMPLO:

A frase “Polícia argumenta que é só um meio de auxílio ao seu trabalho.” irá dar o seguinte resultado:
argumenta={que=1}
polícia={argumenta=1}
seu={trabalho=1}

EXERCÍCIO 22

AUTOR: Professor Tomás Oliveira e Silva

DESCRIÇÃO: Seja n um número inteiro positivo. Neste problema pretende-se encontrar o múltiplo kn de n, o mais pequeno possível da forma 2u + 2v, com u < v.

EXEMPLOS:
- Se n = 4, então u = 2 e v = 3; 3 × 4 = 22 + 23
- Se n = 5, então u = 0 e v = 2; 1 × 5 = 20 + 22
- Se n = 7, então o problema não tem solução
- Se n = 109 então u = 0 e v = 18; 2405 × 109 = 20 + 218
- Se n = 243 então u = 0 e v = 81; 9950006745799417075771 × 243 = 20 + 281

NOTAS:
- Sabe-se que 0 < n < 108

- O valor de n é lido do standard input.
- Se o problema não tiver solução, o programa tem de enviar para o standard output uma linha com o número -1.
- Se o problema tiver solução, o programa tem de enviar para o standard output uma linha com os valores de u e de v.
- O valor de k não é preciso para nada.

Exemplo de standard input (corresponde ao exemplo apresentado acima):
4

Exemplo de standard output (corresponde ao exemplo apresentado acima):
2 3

EXERCÍCIO 23

AUTOR: Professor Tomás Oliveira e Silva

DESCRIÇÃO:

Numa das barracas de jogos da feira de março estão expostos, numa enorme fila ao longo de uma parede, vários objetos. Por cima de cada objeto existe um pequeno alvo. O objetivo do jogo que se joga nesta barraca é derrubar objetos (para ficar com eles), acertando para isso nos alvos. O jogo está sujeito às seguintes regras:

- quando o jogador falha um alvo ou quando já não existem mais objetos o jogo termina, e
- quando o jogador acerta num alvo fica com o objeto correspondente, e os objetos adjacentes (imediatamente à direita e à esquerda), caso ainda estejam em jogo, deixam de estar em jogo.

Um atirador exímio, que nunca falha, pretende ganhar a máxima quantidade de dinheiro possível. (O valor monetário de cada objeto é conhecido.) Em que alvos deverá ele acertar?

Exemplo (com 6 objetos, os casos de teste a sério têm até no máximo 1000 objetos):

- valores monetários dos objetos: 26, 43, 53, 18, 3 e 55
- quantidade máxima de dinheiro que é possível ganhar: 134
- valor monetário dos objetos derrubados (0 significa que o objeto não foi derrubado): 26, 0, 53, 0, 0 e 55

NOTAS:

- O número de objetos é pelo menos 2 e não é maior que 1000.
- O valor monetário de cada objeto, um inteiro, é pelo menos 1 e não é maior que 99.
- Para cada um dos casos de teste a solução é única.
- Os dados do problema são lidos do standard input.
- O programa tem de enviar para o standard output, duas linhas de texto com na primeira linha, a quantidade máxima de dinheiro que é possível ganhar, e na segunda linha, o valor monetário de cada objeto, sendo que esse valor tem se ser substituído por zero caso o objeto não tenha sido derrubado.

EXEMPLO DE INPUT 1:

6

26 43 53 18 3 55

EXEMPLO DE STANDARD OUTPUT 1:
134
26 0 53 0 0 55

EXERCÍCIO 24

AUTOR: Professor Tomás Oliveira e Silva

DESCRIÇÃO:

Dois números primos p e q ímpares distintos formam um par de números primos simétricos se e só se:

mdc(p − 1, q − 1) = |p − q|;

mdc(a, b) é o máximo divisor comum de a e b. Dado um número primo p, neste problema pretende-se que sejam encontrados todos os números primos q que, conjuntamente com p, formam um par simétrico.

EXEMPLO:

- p = 101.
- Valores de q: 97, 103, e 151.

NOTAS:

- Sabe-se que necessariamente 2 < p < 109 (mas o valor fornecido de p pode não ser um número primo).
- O valor de p é lido do standard input.
- Se p não for um número primo, o programa tem de enviar para o standard output uma linha com o número -1.
- Se p for um número primo, o programa tem de enviar para o standard output os valores de q ordenados por ordem crescente, numa única linha e separados por um único espaço. Caso não exista nenhum q, em vez enviar uma linha em branco, o programa tem de enviar uma linha com o número 0.

EXEMPLO DE INPUT 1:

101

EXEMPLO DE OUTPUT 1:

97 103 151

DIFICULDADE

ALTA

EXERCÍCIO 25

AUTOR: Professor Carlos Costa

DESCRIÇÃO:

Dado um ficheiro de texto, o objetivo é codificá-lo ao termo com recurso a um codebook onde só é possível utilizar os seguintes caracteres: 'a'-'z'; 'A' – 'Z'; '0' – '9'. O code book tem de ser guardado no ficheiro de output.

Para este exercício, em termos de performance, é tido em conta o tamanho final do ficheiro. Sendo que um tamanho menor é preferível.

INPUT:

No repositório.

OUTPUT:

No repositório.

EXERCÍCIO 26

AUTOR: Professor Tomás Oliveira Silva

DESCRIÇÃO:

Uma montanhista pretende escalar 10 picos de uma maneira muito peculiar. Para preparar as escaladas, usou imagens de satélite para subdividir a região à volta de cada um dos picos em 40x40 quadrados, e registou a altura média da montanha em cada um desses quadrados.

A montanhista vai escalar cada um dos picos movendo-se de quadrado em quadrado (movimentos diagonais não são permitidos) de modo que a altura do quadrado seguinte é sempre superior à do quadrado onde está.

Encomendou depois um programa de computador que lhe calcula o número total de caminhos possíveis desde um determinado ponto de partida até outro determinado ponto de chegada.

Além de querer saber quantos caminhos existem, a montanhista também quer saber qual é o maior desnível entre quadrados que pode vir a encontrar se escolher à calha um desses caminhos. Foi você que a alpinista contratou.

Exemplo (apenas para 5x5 quadrados, os restantes casos de teste têm todos 40x40 quadrados):

- ponto de partida: 0 1 (linha 0, coluna 1)
- ponto de chegada: 3 4 (linha 3, coluna 4)
- mapa de alturas (5 linhas e 5 colunas)

2	1	2	7	6
1	3	5	8	7
3	5	4	9	20
3	14	15	19	21
30	15	16	17	23

- existem 9 caminhos:

.	1	.	.	.
.	3	.	.	.
.	5	.	.	.
.	14	.	19	21
.	15	16	17	.

desnível 9

.	1	.	.	.
.	3	.	.	.
.	5	.	.	.
.	14	15	19	21
.	.	16	17	.

desnível 9

.	1	.	.	.
.	3	.	.	.
.	5	.	.	.
.	14	15	19	21
.	.	.	.	.

desnível 9

.	1	.	.	.
.	3	5	8	.
.	.	.	9	.
.	.	.	19	21
.	.	.	.	.

desnível 10

.	1	.	.	.
.	3	5	8	.
.	.	.	9	20
.	.	.	.	21
.	.	.	.	.

desnível 11

.	1	2	.	.
.	.	5	8	.
.	.	.	9	.
.	.	.	19	21
.	.	.	.	.

desnível 10

.	1	2	.	.
.	.	5	8	.
.	.	.	9	20
.	.	.	.	21
.	.	.	.	.

.	1	2	7	.
.	.	.	8	.
.	.	.	9	.
.	.	.	19	21
.	.	.	.	.

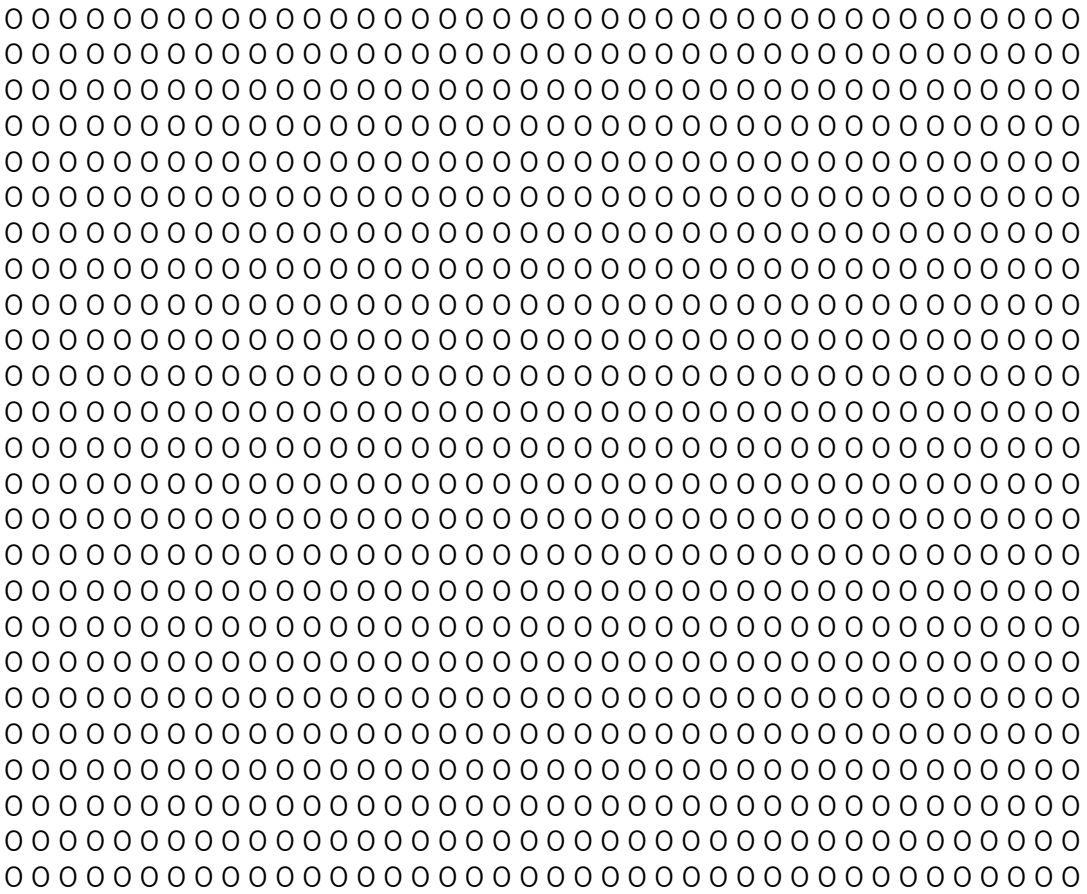
.	1	2	7	.
.	.	.	8	.
.	.	.	9	20
.	.	.	.	21
.	.	.	.	.

Notas:

- Se o número de caminhos for maior que 10^18, o programa deve indicar que o número de caminhos é 10^18.
- Se não houver nenhum caminho, o programa deve indicar que o número de caminhos é 0 e que o maior desnível também é 0.
- Os dados do problema são lidos do standard input.
- O programa tem de enviar para o standard output o número de caminhos numa linha, e o maior desnível noutra linha (sem espaços extra).
- As alturas (números inteiros) são maiores ou iguais a zero e menores, ou iguais a 10000.

EXEMPLO DE INPUT:

```
0 1
3 4
2 1 2 7 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 3 5 8 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 5 4 9 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 14 15 19 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
30 15 16 17 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



EXEMPLO DE OUTPUT:

9
11

EXERCÍCIO 27

DESCRIÇÃO:

O Carlos tem uma tarefa simples na mão: contar o número de divisores de um determinado número positivo *M*. Por exemplo, o número 60 possui 12 divisores 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 e 60. Embora seja uma tarefa muito fácil, encontrar o contrário não é fácil. Se receber o número de divisores *D* de um número positivo desconhecido *M*, não é muito fácil encontrar *M* e em todos os casos há mais do que uma solução. As tarefas chatas e fáceis dadas pelos seus professores não mantêm Carlos interessado por muito tempo. Então, ele está a tentar resolver esta tarefa bastante difícil. Consegues ajudá-lo?

INPUT:

O input contém várias linhas de entrada. Cada linha contém um número inteiro D ($0 < D \leq 5000$).

OUTPUT:

Para cada input produza uma linha de output. Esta linha deve conter um número positivo M menor que $(10^{15}+1)$ cujo número dos divisores é exatamente *D*. Se não houver esse número *M* menor que $(10^{15}+1)$ cujo número total de divisores é *D*, imprima a palavra 'Impossible'' sem as aspas. Se houver mais de um valor possível de *M* dentro do intervalo especificado, imprima o menor.

EXEMPLO DE INPUT 1:

3

EXEMPLO DE INPUT 2:

60

EXEMPLO DE OUTPUT 1:

4

EXEMPLO DE OUTPUT 2:

5040

EXERCÍCIO 28

AUTOR: Professor Mário Antunes

DESCRIÇÃO:

A sequência de collatz é definida por:
n → n/2 (n é par)
n → 3n + 1 (n é ímpar)
E embora não esteja provado, é admitido que qualquer que seja o número positivo inicial o resultado da aplicação da sequência resultará eventualmente no valor 1. A sequência só é válida para valores positivos [1, ∞[
Para 13: 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1
Queremos uma aplicação que dado um valor limite retorne o valor que gera a maior sequência e o tamanho da sequência.

Considerações: o input será o limite da pesquisa. O output será o número com maior sequência e o tamanho da sequência.

INPUT:

O input consiste numa sequência de pares inteiros *n* e *p* com cada número inteiro numa linha por si só. Para todos esses pares $1 \leq n \leq 200$, $1 \leq p < 10101$ e existe um número inteiro *k*, $1 \leq k \leq 109$ tal que $k^n = p$.

OUTPUT:

Para cada par inteiro *n* e *p*, o valor np deve ser impresso, ou seja, o número k tal que $k^n = p$.

EXEMPLO DE INPUT 1:

100

EXEMPLO DE INPUT 2:

1000000

EXEMPLO DE OUTPUT 1:

97 118

EXEMPLO DE OUTPUT 2:

837799 524

EXERCÍCIO 29

DESCRIÇÃO:

Na teoria dos grafos, um conjunto de arestas correspondentes ou independentes definidas num grafo $G = (V, \mathcal{E})$ é um conjunto de arestas $M \subseteq \mathcal{E}$ de modo a que não haja duas arestas no M correspondente que compartilhem um vértice comum.
Recentemente, viste nas notícias que "O Prêmio Sveriges Riksbank em Ciências Económicas em Memória de Alfred Nobel"(informalmente, o Prêmio Nobel de Economia) em 2012 foi concedido a Alvin E. Roth e Lloyd S. Shapley por, entre outras coisas, o algoritmo dos mesmos para encontrar uma correspondência que satisfaça certos critérios num grafo bipartido. Como também ouviste que as correspondências nos grafos de ciclo têm aplicações em química, os teus pensamentos concentram-se num plano para um futuro bonito, onde suas compras de Natal são mais luxuosas do que nunca!
O grafo de ciclo, C_n, $n \geq 3$, é um grafo simples e não direcionado, no conjunto de vértices {1,. . . , n}, com conjunto de arestas

$\mathcal{E}(C_n) = \{\{a, b\} \mid |a - b| \text{ (igual de 3 linhas) } 1 \bmod n\}$. É 2-regular e contém n arestas. Os gráficos C_3, C_4, C_5, e C_6 estão representados na Figura 1.

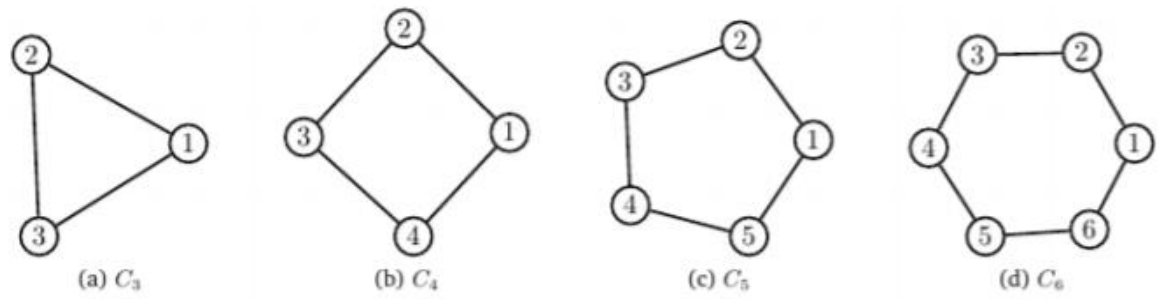


Figura 1: Os grafos C_3, C_4, C_5, e C_6.

O teu primeiro passo para a fama do Prêmio Nobel é poder calcular o número de correspondências no gráfico de ciclo C_n. Na Figura 2 estão representadas as sete combinações do gráfico C_4.

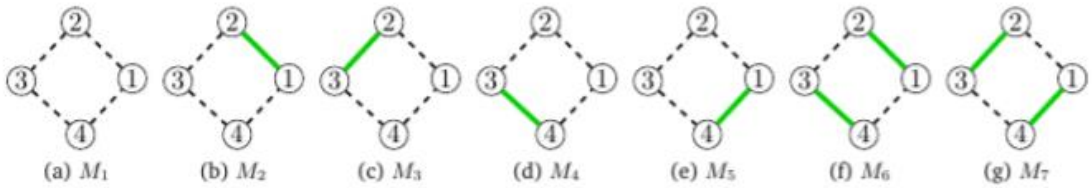


Figura 2: As combinações de C4. As arestas que fazem parte da respectiva correspondência são coloridas a verde, enquanto as arestas deixadas de fora da

correspondência são tracejadas. $M1$ = 0, $M2$ = {{2, 1}}, $M3$ = {{3, 2}}, $M4$ = {{4, 3}}, $M5$ = {{1, 4}}, $M6$ = {{2, 1}, {4, 3}} e $M7$ = {{3, 2}, {1, 4}}.

INPUT:

Para cada caso de teste, uma única linha contendo um número inteiro positivo: n, com $3 \leq n \leq 10000$.

OUTPUT:

Para cada caso de teste, uma linha contendo o número de correspondências em Cn.

EXEMPLO DE INPUT 1:

3

EXEMPLO DE INPUT 2:

4

EXEMPLO DE OUTPUT 1:

4

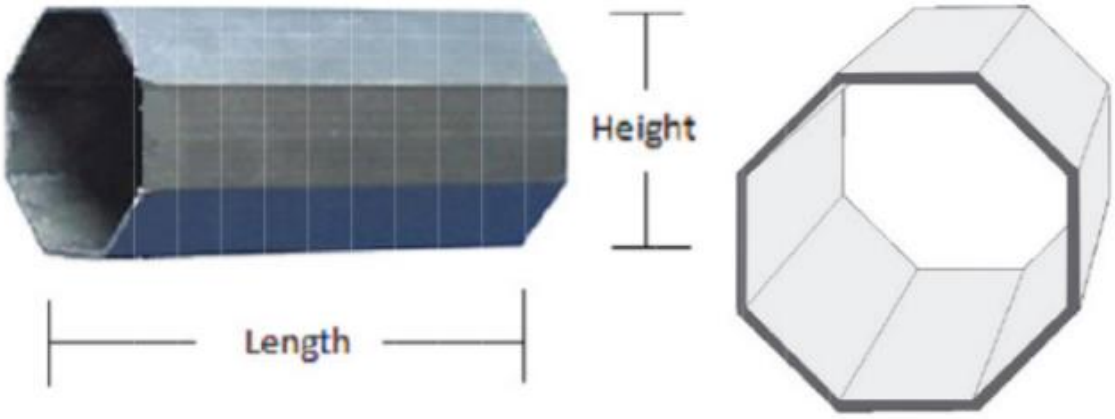
EXEMPLO DE OUTPUT 2:

7

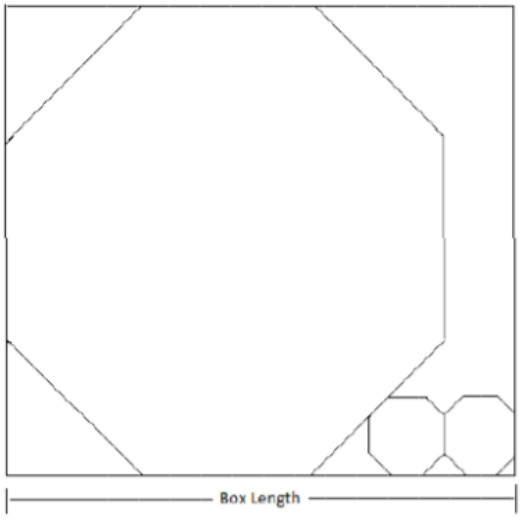
EXERCÍCIO 30

DESCRIÇÃO:

Estamos a fazer uma caixa para embalar alguns tubos octogonais regulares. A caixa tem 8 “pés” de largura e o comprimento de cada tubo é também 8 “pés”. Mas eles têm diferentes alturas de secção transversal. Um tubo octogonal regular possui altura e largura da secção transversal igual. Se não estiveres familiarizado com tubos octogonais, vê as figuras em baixo.



Estes tubos serão colocados na caixa de modo a que os seus comprimentos estejam alinhados com a largura da caixa e um dos oito lados deve tocar completamente no chão da caixa, o que significa que não podemos empilhá-los um em do outro. A figura a seguir mostra um exemplo específico de embalagem, onde três tubos foram embalados na caixa.



Se receberes as alturas de secção transversal dos tubos que queremos embalar na caixa, consegues calcular o volume mínimo possível para a caixa? Podes mudar a ordem dos tubos, mas tens que seguir as regras para as colocar na caixa. Podes assumir, que o número de tubos é no máximo 8, a altura de cada tubo não é maior que 100 “pés” e pode haver no máximo 100 casos de teste.

INPUT:

Para cada caso de teste, teremos um número inteiro N ($1 \leq N \leq 8$) seguido por N números inteiros H_i ($1 \leq H_j \leq 100000$) em que N é o número de tubos e H_i é a altura da seção transversal do i-ésimo tubo.

OUTPUT:

Para cada caso de teste, imprima o volume desejado da caixa, conforme mostrado na amostra de output.
Nota: Se o output for um número decimal arrende o mesmo para 6 casas decimais.

EXEMPLO DE INPUT 1:

3 5 5 5

EXEMPLO DE INPUT 2:

6 10 2 2 9 4 6

EXEMPLO DE OUTPUT 1:

600

EXEMPLO DE OUTPUT 2:

2331.370850

EXERCÍCIO 31

DESCRIÇÃO:

Dadas 2 equações nas variáveis x e y, resolva x e y.

INPUT:

Cada caso de teste consiste em duas equações, cada uma numa linha separada. Uma equação consiste em dois ou mais termos separados por operadores de adição, subtração ou igualdade.
Um termo é um número inteiro ou um nome de variável (x ou y) opcionalmente precedido por um sinal de menos ou um coeficiente inteiro.
Existe exatamente um operador de igualdade. Todos os operadores são cercados por espaços e não há espaços dentro dos termos.

OUTPUT:

Para cada caso, imprime duas linhas, fornecendo os valores de x e y como racionais nos termos mais simples. Se x ou y não tiverem um valor racional único
de modo a que ambas as equações sejam válidas, imprima "don't know".

EXEMPLO DE INPUT 1:

2x + 3y = x
5 = x + y + 3

EXEMPLO DE INPUT 2:

2x + 3y = 0
10x = -15y

EXEMPLO DE OUTPUT 1:

3
-1

EXEMPLO DE OUTPUT 2:

don't know
don't know

EXERCÍCIO 32

DESCRIÇÃO:

Queremos ter um ótimo SWERC no Porto este ano e enfrentamos esse desafio de várias maneiras.
Até o enquadramos como um problema de adição de palavras, semelhante ao clássico SEND+MORE=MONEY, em que cada letra representa um único dígito (0, 1, 2, ..., 8, 9) que fazem com que a operação aritmética seja correta. Dentro da adição de palavras letras diferentes não podem ser atribuídas ao mesmo dígito e a letra mais à esquerda numa palavra não pode ser zero (0). Em particular, um termo de letra única não pode ser zero.

Para resolver este problema de adição de palavras, tivemos que encontrar dígitos positivos para G, S e P e dígitos para R, E, A, T, W, C, O, de modo a que cada letra tenha um dígito diferente e a soma esteja correta. Acontece que, ao contrário do clássico SEND+MORE=MONEY, que possui uma única solução,
GREAT + SWERC = PORTO
tem seis soluções.

T = 7, E = 3, W = 9, G = 1, A = 0, P = 4, S = 2, C = 8, R = 6, O = 5
T = 7, E = 3, W = 9, G = 2, A = 0, P = 4, S = 1, C = 8, R = 6, O = 5
T = 8, E = 5, W = 1, G = 3, A = 7, P = 9, S = 6, C = 4, R = 0, O = 2
T = 8, E = 5, W = 1, G = 6, A = 7, P = 9, S = 3, C = 4, R = 0, O = 2
T = 9, E = 5, W = 2, G = 1, A = 8, P = 7, S = 6, C = 4, R = 0, O = 3
T = 9, E = 5, W = 2, G = 6, A = 8, P = 7, S = 1, C = 4, R = 0, O = 3

Ter mais de uma solução não torna o GREAT + SWERC = PORTO um bom problema para resolver manualmente, mas ainda é bastante fácil para um programador.

Dado um problema de adição de palavras, calcule o número de soluções (possivelmente zero).

INPUT:

Cada caso de teste é descrito abaixo:

Uma linha com um número inteiro *n*, seguida por *n* linhas contendo uma palavra, cada uma com comprimento máximo de 10 letras. As primeiras *n - 1* palavras são os termos a serem adicionados e a última linha é o resultado. As palavras contêm apenas letras maiúsculas. Se as palavras tiverem comprimentos diferentes, elas devem ser interpretadas como alinhadas à direita.
Por exemplo, no problema e SEND+MORE=MONEY, o D da primeira palavra e E da segunda alinham com o Y da palavra final. Também se pode assumir que o tamanho da última palavra é maior ou igual ao tamanho máximo das palavras anteriores e, além disso, no máximo dez letras distintas estão envolvidas num problema de palavras.

Restrições:
$3 \leq n \leq 10$
Cada palavra tem no máximo 10 símbolos (letras maiúsculas).
Um problema de palavras tem no máximo 10 letras distintas.

OUTPUT:

Para cada caso de teste, imprime uma única linha com um número inteiro: o número de soluções do problema da adição de palavras dado como input.

EXEMPLO DE INPUT 1:

3
GREAT
SWERC
PORTO

EXEMPLO DE INPUT 2:

3
SEND
MORE
MONEY

EXEMPLO DE OUTPUT 1:

6

EXEMPLO DE OUTPUT 2:

1

EXERCÍCIO 33

DESCRIÇÃO:

A Sara acaba de atingir o último nível do jogo de computador 'magicRings". Ela usou a maioria dos seus super poderes nos níveis anteriores ao derrotar os monstros. Felizmente, não há muitos monstros neste nível que precisam de ser enfrentado. O primeiro passo deste nível é identificar os locais dos monstros. Depois de identificar os locais, Sara tem que largar os super poderes um depois do outro no chão, em locais diferentes, para atacar os monstros. Uma super poder, quando largado em determinado ponto, cria um anel mágico de certo raio. Qualquer monstro que esteja dentro da periferia deste anel será ferido. Os raios
dos anéis gerados pelos super poderes são todos iguais, mas o seu valor está no controlo da Sara. Sara escolhe o menor raio possível. Todos os anéis terão esse raio específico que a Sara escolhe.
Dada a localização de N monstros e o número de super poderes que Sara tem à sua disposição, consegues descobrir o raio mínimo dos anéis mágicos que seria suficiente para ferir todos os monstros?

INPUT:

Cada caso de teste começa com dois números inteiros N ($0 < N < 19$) e ($0 < K \leq N$). N representa o número de monstros e K representa o número de super poderes em mãos. A próxima linha contém as coordenadas dos N monstros no formato *x1 y1 x2 y2 xn yn*. Todas as coordenadas são números inteiros no intervalo [0, 10000].

OUTPUT:

Para cada caso imprime o raio mínimo dos anéis mágicos arredondados para 2 casas decimais.

EXEMPLO DE INPUT 1:

2 1
0 0 10 0

EXEMPLO DE INPUT 2:

6 2
0 0 1 1 2 2 10 10 10 11 10 12

EXEMPLO DE OUTPUT 1:

5.00

EXEMPLO DE OUTPUT 2:

1.41

EXERCÍCIO 34

DESCRIÇÃO:

ACRush é bastante famoso na Supercoder. A Supercoder é uma empresa profissional que organiza concursos algorítmicos online e classifica as pessoas com base nesses concursos. No ranking dos concursos algorítmicos da Supercoder, o ACRush está em terceiro. Recentemente, ele fez algumas análises sobre o seu histórico de classificação no concurso de algoritmos da Supercoder. Na Supercoder, um concurso de algorítmico é denominado como "Single Round Tournament" (SRT). Após a conclusão de cada SRT, a classificação de um concorrente é atualizada de acordo com o seu desempenho relativo. O ACRush colecionou todas essas informações de classificação e, usando essas, criou um gráfico de linhas. Para deixar as coisas mais claras, consideremos a tabela a seguir como informações de classificação.

SRT	Rating
320	3
306	1
401	3
325	4
393	5
380	2

A partir desta tabela, vemos que o seu primeiro SRT foi SRT # 306, e a classificação após essa SRT foi 1; portanto, ele marcou o ponto (1, 1) como r1 no papel milimétrico, o seu segundo SRT foi o SRT # 320 e a classificação após esse SRT era 3, então ele marcou (2, 3) como r2, depois adicionou r1 com r2 por uma linha reta e assim por diante. Em geral, para o seu i-ésimo SRT, ele marcou o ponto (i, classificação após o i-ésimo SRT) por *ri*. Depois de marcar todos os pontos, ele adicionará o ponto ri com *ri-1* por linhas retas, para todos $1 < i \leq N$, onde N é o número total de SRTs que ele jogou. Para uma ideia melhor, veja a figura 1:

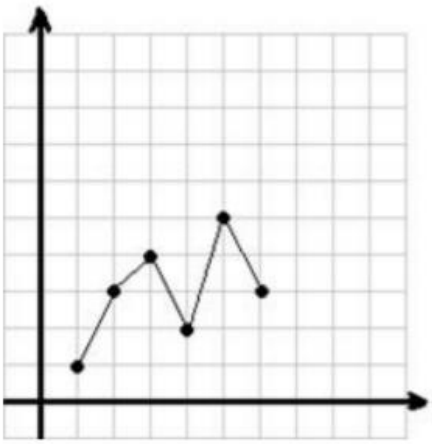


Fig 1: Gráficos de linhas a considerar todos os SRTs

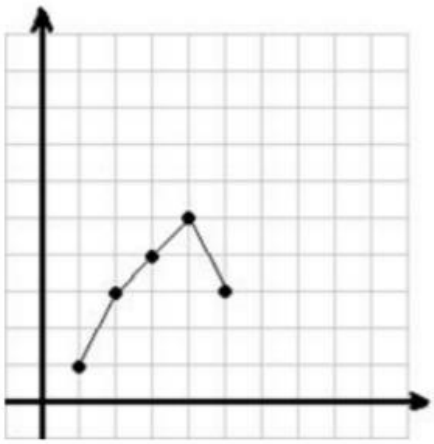


Fig 2: Gráficos de linhas a ignorar SRT#380

Depois de desenhar o gráfico de linhas, ele ficou muito interessado no número de picos. Existem dois tipos de picos num gráfico de linhas: 1) Pico superior e 2) Pico inferior. O Pico superior é o ponto num gráfico de linhas cujo ponto anterior e próximo tem coordenadas y menores e o pico mais baixo é aquele cujo ponto anterior e próximo tem coordenadas y maiores. Por exemplo, o número total de picos na figura 1 é 3. Dois deles, pico superior, que são (3, 4) e (5, 5), e um deles é o pico inferior, que é (4, 2). O ACRush observou que, ignorando o SRT # 380, o seu gráfico de linhas tornará-se-á como a figura 2, em que o número de picos é de apenas 1. Ao observar isso, ficou mais curioso. Agora quer saber, ignorando 0 ou mais SRTs quantos gráficos de linhas distintos com K picos são possíveis. O ACRush chama esses gráficos "gráficos de linhas *K-peak*". Num "gráfico de linhas K-peak", ele não permite que dois pontos consecutivos tenham a mesma coordenada y.

INPUT:

Cada caso de teste começa com uma linha com dois números inteiros N ($1 \leq N \leq 10000$) e K ($0 \leq K \leq 50$). Cada uma das próximas N linhas contém dois números inteiros *SRT* ($1 \leq SRT \leq 1000000000$) e *Rating* ($1 \leq Rating \leq 1000000000$). Todos os números SRT serão distintos.

OUTPUT:

Para cada caso de teste gera uma única linha que contenha W sendo W o número de "gráficos de linhas *K-peak*" distintos módulo 1000000.

EXEMPLO DE INPUT 1:

6 1

320 3
306 1
401 3
325 4
393 5
380 2

EXEMPLO DE INPUT 2:

4 1
101 3
102 2
103 2
104 4

EXEMPLO DE OUTPUT 1:

20

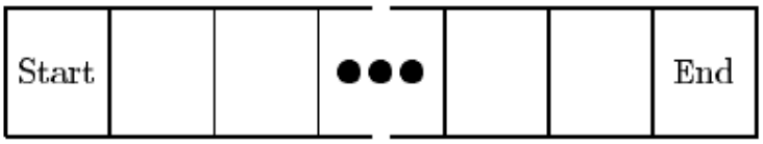
EXEMPLO DE OUTPUT 2:

1

DIFICULDADE EXTREMA

EXERCÍCIO 35

DESCRIÇÃO:



Foi criado um novo jogo destinado a apostas.

Inicialmente, é colocada uma ficha no quadrado “Start”. O jogador tenta, ao longo de turnos, chegar ao quadrado “End”, acabando assim com o jogo. Em cada turno, é atirada ao ar uma moeda: se a moeda calhar “Cara”, a ficha avança uma casa para a direita, e se calhar “Coroa”, avança duas casas para a direita (a não ser que a ficha esteja a uma casa do fim, sendo que se move para o quadrado “End”). A partir daí, qualquer instrução no quadrado onde a moeda cai tem que ser seguida. Cada instrução é uma das seguintes:

- Andar para a esquerda n quadrados (n é um inteiro positivo)
- Andar para a esquerda n quadrados (n é um inteiro positivo)
- Perdes o turno
- Não há instrução

Após seguir a instrução, o turno acaba e um novo turno começa. Assim que a ficha cair no quadrado originado pela instrução, não avança mais.

As apostas são feitas da seguinte forma: é dado um tabuleiro com layout e um inteiro T, deves apostar se o jogo vai acabar em T rondas ou não.

INPUT:

Cada instância consistirá em duas linhas: a primeira conterá dois números inteiros **m** e **T** ($1 \leq m \leq 50, 1 \leq T \leq 40$), em que m é o tamanho do quadro excluindo os quadrados ”Start” e “End” e T é o número alvo de turnos.
A próxima linha conterá instruções para cada um dos m quadrados interiores no quadro. As instruções para os quadrados serão separadas por um único espaço, e uma instrução será uma das seguintes: '+ n', '-n', 'L' ou '0' (o dígito zero). O primeiro indica um movimento à direita de n quadrados, o segundo um movimento à esquerda de n quadrados, o terceiro um quadrado de perder o turno e o quarto indica nenhuma instrução para o quadrado. Nenhum movimento da direita ou da esquerda o levará para fora do tabuleiro.

OUTPUT:

O output para cada instância do problema consistirá em uma linha,

Bet for. x.xxxx

se você acha que há mais de 50% de chance de o jogo terminar em T ou menos turnos, ou
Bet against. x.xxxx

Se você acha que há menos de 50% de chance de o jogo terminar em T ou menos turnos, ou
Push. 0,5000

Caso contrário, onde 'x.xxxx' é a probabilidade de o jogo terminar em T ou menos turnos, arredondado para 4 casas decimais. (Observe que, devido ao arredondamento da probabilidade calculada para exibição, uma probabilidade de
"0,5000" pode aparecer após a mensagem "Bet for." ou "Bet against.")

EXEMPLO DE INPUT 1:

4 4
0 0 0 0

EXEMPLO DE INPUT 2:

3 3
0 -1 L

EXEMPLO DE OUTPUT 1:

Bet for. 0.9375

EXEMPLO DE OUTPUT 2:

Bet against. 0.0000

EXERCÍCIO 36

DESCRIÇÃO:

O país em que moras é na verdade uma terra de rios. Quase todas as cidades do país estão cercadas pelos rios. Felizmente, existem muitas pontes no país que conectam as cidades. De repente o país está sob ataque e a oposição está a tentar destruir as pontes para isolar as cidades. Tu, como comandante de um setor, agora desejas saber o status atual do setor. O setor consiste em N cidades. Tu pediste ao teu assistente para recolher as informações de conectividade atuais do setor. Mas ele só conseguia obter d_i para cada uma das cidades do setor. Aqui d_i representa o número de cidades diferentes às quais i está diretamente conectado. Observe que uma cidade não é considerada conectada consigo mesma. Agora, tu desejas desenhar um grafo que seja consistente com as informações. Também queres atribuir a cada uma das cidades uma cor tal que as cidades conectadas diretamente não tenham a mesma cor. Mas tens apenas duas cores para usar. Esse grafo pode ser desenhado?

INPUT:

Cada caso de teste começa com uma linha que contém um número inteiro N ($2 \leq N \leq 20$). A próxima linha contém N números inteiros d_i ($0 \leq d_i \leq N$).

OUTPUT:

Para cada caso, imprima 'YES', se for possível desenhar um grafo ou 'NO', se não for.

EXEMPLO DE INPUT 1:

```
4
3 1 3 3
```

EXEMPLO DE INPUT 2:

```
5
1 2 1 1 3
```

EXEMPLO DE OUTPUT 1:

```
NO
```

EXEMPLO DE OUTPUT 2:

```
YES
```

EXERCÍCIO 37

DESCRIÇÃO:

O professor Xing está a trabalhar num produto chamado Balance por alguns dias. A nova balança inventada por ele pode especificar com precisão a diferença entre dois pesos. Como resultado, é possível pesar qualquer coisa usando apenas um peso padrão.
O professor quer testar o novo Balance. Então ele coletou algumas esferas metálicas de diferentes pesos.
Agora ele mantém uma esfera de um lado da balança e, uma a uma, coloca outras esferas no outro lado para tirar a diferença de peso entre duas esferas. Quando a diferença se torna negativa, ele ignora, caso contrário, ele escreve a diferença num papel.
Ele realiza esse processo com cada esfera individualmente. Agora, usando os dados o professor deseja testar a correção da nova balança. A ideia dele é simples. Se a, b, c (a <b <c) são três pesos, então (c - a) - (b - a) = c - b.
Mas ele encontrou um problema! O professor esqueceu-se da ordem correta do seu esquema de pesos. Ele está à procura da tua ajuda.
A tua tarefa é encontrar um peso para cada esfera, de modo que todas as diferenças possíveis de peso correspondam à leitura do professor.

INPUT:

A entrada começa com um número inteiro, N (3 ≤ N ≤ 50). Haverá (N*N (N −1)) / 2 números inteiros positivos (menor que 10000) indicando as leituras da balança. A entrada será encerrada por EOF.

OUTPUT:

Se obtiveres solução em cada caso deves imprimir N números inteiros numa única linha que indica o peso de cada esfera em ordem crescente de peso. Pode haver infinitas soluções para o problema porque se for adicionado um offset d a cada peso, a diferença de peso não será alterada. Então o primeiro o peso deve ser zero. Outra coisa importante é que cada solução tem uma simétrica que também é uma solução.
Como no output 1 a imagem de (0 2 3) é (0 1 3), que também é uma solução. Tens que mostrar o primeiro.
Se nenhuma solução for encontrada, basta imprimir "Incorrect Balance".

EXEMPLO DE INPUT 1:

3
1 2 3

EXEMPLO DE INPUT 2:

3
1 2 2

EXEMPLO DE OUTPUT 1:

0 2 3

EXEMPLO DE OUTPUT 2:

Incorrect Balance.

EXERCÍCIO 38

DESCRIÇÃO:

O jogo de pedras do Miguel é o seguinte: dois jogadores iniciam o jogo com uma pilha de n pedras.
Eles tiram pedras de uma pilha na sua vez e todos os turnos eles pegam em pelo menos uma pedra.

Quem vai primeiro pode levar no máximo n - 1 pedras para o seu primeiro turno. A partir de esse turno, um jogador pode levar no máximo k vezes quantas pedras o seu oponente levou na sua última vez. Por exemplo, se um jogador pega m pedras no seu turno, então o outro
jogador pode tirar no máximo k × m pedras na próxima vez. O jogador que pegar a última pedra vence o jogo.

Suponha que esses dois jogadores sempre realizam as melhores jogadas e nunca cometem erros, o teu trabalho é descobrir quem definitivamente ganhará o jogo.

INPUT:

Cada caso de teste é uma linha que consiste em dois números n e k. ($2 \leq n \leq 108$, $1 \leq k \leq 105$).

OUTPUT:

Se o primeiro jogador pode garantir uma vitória, imprima o número mínimo de pedras que ele deve levar no seu primeiro turno.
Caso contrário, imprima 'lose'.

Dica:
Quando k = 1, o primeiro jogador perderá definitivamente se a quantidade inicial de pedras estiver no set {2, 4, 8, 16, 32,. . .}. Vamos chamar a este conjunto de "primeiro jogador a perder".
Quando k = 2, o conjunto de derrotas no primeiro jogador é {2, 3, 5, 8, 13, 21, 34, 57,. . .}, que passa a ser a sequência de Fibonacci a partir de 2.

EXEMPLO DE INPUT 1:

16 1

EXEMPLO DE INPUT 2:

11 1

EXEMPLO DE OUTPUT 1:

1

EXEMPLO DE OUTPUT 2:

3

EXERCÍCIO 39

DESCRIÇÃO:

Um problema complexo em química é a tarefa de equilibrar o número de átomos numa equação química.
Os químicos obedecem a essas regras quando apresentam equações químicas:

- O nome de cada elemento é abreviado por no máximo duas letras. A primeira letra está sempre em maiúscula e a segunda letra, se existir, é uma letra minúscula (por exemplo, o cálcio é representado por Ca, Oxigênio por O e Cloro por Cl).
- Cada molécula é composta por um número de átomos. Para representar uma molécula, concatenamos os nomes abreviados dos seus átomos compostos. Por exemplo, NaCl representa cloreto de sódio. Cada nome do átomo pode ser seguido por um número de frequência. Por exemplo, cloreto de cálcio CaCl2 consiste num átomo de cálcio e dois átomos de cloro. Se a frequência não for dada, éassumido como 1 (então HCl é equivalente a H1Cl1). Por uma questão de simplicidade, podemos assumir que a frequência da ocorrência de um átomo é no máximo 9 (portanto, não temos C11H22O11 no input). Observa que pode haver várias ocorrências do mesmo átomo na molécula fórmula, como o átomo H em CH3COOH.
- Em reações químicas comuns, várias moléculas se combinam e resultam em várias outras moléculas. Por exemplo, uma amostra bem conhecida de neutralização é:

  2HCl + CaO2H2 → CaCl2 + 2H2O

  Isso significa duas moléculas de ácido cloro-hídrico (HCl) com uma molécula de hidróxido de cálcio resulta numa molécula de cloreto de cálcio (CaCl2) e duas moléculas de água.

- Em todas as reações químicas, o número total de cada átomo no lado direito da equação é igual ao número total desse átomo no lado esquerdo.

A tua tarefa é escrever um programa para encontrar os coeficientes apropriados x1, x2,. . . , xM e y1, y2,. . . , yN para equilibrar uma equação (desequilibrada) como

A1 + A2 + A3 + + AM → B1 + B2 + B3 + + + BN

Da seguinte maneira:

x1A1 + x2A2 + x3A3 + xMAM → y1B1 + y2B2 + y3B3 + · · + yN BN

INPUT:

Cada caso de teste consiste numa única linha contendo uma expressão como:

A1 + A2 + A3 + + AM = B1 + B2 + B3 + + + BN

Cada Ai ou Bi é uma fórmula de uma molécula de acordo com as regras indicadas nos pontos 1 e 2.
As equações de entrada são dadas de uma maneira que, se puderem ser equilibradas, os coeficientes xi e yi podem estar no intervalo de 1 a 9. Existem menos de 10 moléculas e menos de 10 tipos diferentes de átomos, numa dada equação. Além disso, podes assumir que as moléculas não contêm mais que 3 tipos diferentes de átomos. Também podes assumir que não há caracteres em branco no ficheiro de entrada e que o comprimento máximo das linhas de entrada é de 200 caracteres.

OUTPUT:

A saída tem que ter a lista de coeficientes necessários, separados por espaços em branco, na seguinte ordem:

x1 x2. . . xM y1 y2. . . yN

Os coeficientes devem ser números inteiros no intervalo de 1 a 9. Obviamente, pode haver mais de uma resposta para um caso de teste. Em tais situações, imprima a resposta que minimiza o número:

x1x2. . . xMy1y2. . . yN

Se a equação for impossível de equilibrar, o output deve ser "IMPOSSIBLE".

EXEMPLO DE INPUT 1:

HCl+CaO2H2=CaCl2+H2O

EXEMPLO DE INPUT 2:

HCl+H2SO4=NaCl

EXEMPLO DE OUTPUT 1:

2 1 1 2

EXEMPLO DE OUTPUT 2:

IMPOSSIBLE

EXERCÍCIO 40

AUTOR: ITSector

DESCRIÇÃO:

Decifradas mensagens de uma fação inimiga, numa hipotética guerra, obtivemos como resultado uma sequência de caracteres, que te apresentamos de seguida, nos inputs das mensagens 1 e 2.

Cada caracter ou conjunto de caracteres poderão constituir uma palavra completa ou serem parte constituinte de uma palavra.

Após as submeteres ao algoritmo que deverás construir, obterás as seguintes frases finais, como indicado nos outputs das mensagens 1 e 2.

Obs: todos os campos fornecidos são importantes para chegares à solução final.

EXEMPLO DE INPUT 1:

```
1|Além
1|avançar, se
1|Se,nã,avançar
1|consegues,alterar,ok
-1|ler
2|ler,outra,vez
-1|corretamente
1|anterior,mensagem
0|corretamente,a
1|erro,amanhã
0|esta,telefonema
0|mensagem
2|podes,não
1|significa,que,nada
1|estás,longe
0|nã
0|não
0|no
3|mau,péssimo
-1|bom,fim
0|estrada
0|fora
0|caminho
```

EXEMPLO DE INPUT 2:

```
1|Vamos
```

1|Vai, inalar
1|Vamos,inalar,um
0|o
1|um,atado
-1|iniciar,um
1|ataque
1|como
0|conto
0|com
1|amas
1|armas,química
0|químicas
1|hoje.
1|A
1|embrolhada
0|emboscada
2|se,há
1|feita
-1|será,feita
1|porque
1|norte.
0|por,norte

EXEMPLO DE OUTPUT 1:

Se consegues ler corretamente esta mensagem significa que estás no bom caminho

EXEMPLO DE OUTPUT 2:

Vamos iniciar um ataque com armas químicas hoje. A emboscada será feita por norte

THINK TWICE, BE PRECISE!