# Math

**Constantes Matemáticas:**

* math.pi: Retorna o valor de π (pi).
* math.e: Retorna o valor da constante de Euler (e).
* math.inf: Representa o infinito positivo.
* math.nan: Representa um valor de "não é um número" (NaN).

**Funções Trigonométricas:**

* math.sin(x): Retorna o seno de x (em radianos).
* math.cos(x): Retorna o cosseno de x (em radianos).
* math.tan(x): Retorna a tangente de x (em radianos).
* math.asin(x): Retorna o arco seno de x (em radianos).
* math.acos(x): Retorna o arco cosseno de x (em radianos).
* math.atan(x): Retorna o arco tangente de x (em radianos).
* math.atan2(y, x): Retorna o arco tangente de y / x (em radianos), considerando os sinais de y e x.

**Funções Hiperbólicas:**

* math.sinh(x): Retorna o seno hiperbólico de x.
* math.cosh(x): Retorna o cosseno hiperbólico de x.
* math.tanh(x): Retorna a tangente hiperbólica de x.
* math.asinh(x): Retorna o arco seno hiperbólico de x.
* math.acosh(x): Retorna o arco cosseno hiperbólico de x.
* math.atanh(x): Retorna o arco tangente hiperbólico de x.

**Funções Exponenciais e Logarítmicas:**

* math.exp(x): Retorna exe^xex.
* math.log(x[, base]): Retorna o logaritmo natural de x (base eee). Se a base for especificada, retorna o logaritmo de x na base especificada.
* math.log2(x): Retorna o logaritmo de x na base 2.
* math.log10(x): Retorna o logaritmo de x na base 10.

**Outras Funções:**

* math.sqrt(x): Retorna a raiz quadrada de x.
* math.pow(x, y): Retorna xyx^yxy.
* math.ceil(x): Retorna o menor inteiro maior ou igual a x.
* math.floor(x): Retorna o maior inteiro menor ou igual a x.
* math.factorial(x): Retorna o fatorial de x.
* math.degrees(x): Converte o ângulo x de radianos para graus.
* math.radians(x): Converte o ângulo x de graus para radianos.

# 

# Grafos

Sam encontrou vários mapas que deveriam levar a baús de obsidiana, mas alguns estão com erros. Seu trabalho é criar um programa que verifica se um mapa é válido ou não.

Características dos mapas:

* Ponto de partida: canto superior esquerdo.
* Formato: retangular.
* Símbolos: espaço atravessável, flechas (direita, esquerda, cima, baixo) e baú.

Entrada:

* Primeira linha: inteiro positivo x (< 100) indicando a largura do mapa.
* Segunda linha: inteiro positivo y (< 100) indicando a altura do mapa.
* Linhas seguintes: caracteres representando o mapa (>, <, v, ^, ., \*).

Saída:

* ! se o mapa é inválido.
* \* se o mapa é válido.

**c = int(input())**

**l = int(input())**

**m = []**

**for i in range(l):**

**m.append([e for e in input()])**

**t = False**

**i = j = 0**

**a = '!'**

**while True:**

**if m[i][j] == '.': m[i][j] = a**

**if m[i][j] == '\*':**

**t = True**

**break**

**elif m[i][j] == '!': break**

**elif m[i][j] == '>':**

**if j < c-1:**

**a = '>'**

**m[i][j] = '!'**

**j += 1**

**else: break**

**elif m[i][j] == '<':**

**if 0 < j:**

**a = '<'**

**m[i][j] = '!'**

**j -= 1**

**else: break**

**elif m[i][j] == '^':**

**if 0 < i:**

**a = '^'**

**m[i][j] = '!'**

**i -= 1**

**else: break**

**elif m[i][j] == 'v':**

**if i < l-1:**

**a = 'v'**

**m[i][j] = '!'**

**i += 1**

**else: break**

**print(m[i][j])**

# Geometria Computacional

**1 - Flores de Fogo**

**Contexto**

Nos dias de hoje, muitos jovens reconhecem uma flor de fogo por causa de um famoso jogo de videogame onde o protagonista ganha superpoderes ao tocá-la, permitindo-lhe atirar bolas de fogo.

Historicamente, flores de fogo aparecem na mitologia polonesa como flores místicas de grande poder, guardadas por espíritos malignos. Elas brilhavam intensamente e concediam habilidades como ler mentes, encontrar tesouros e repelir males.

Para obter uma flor de fogo, uma pessoa devia procurá-la em uma floresta na véspera do Noc Kupały, antes da meia-noite. À meia-noite, a flor floresceria, e a pessoa precisava desenhar um círculo em volta dela. No entanto, espíritos malignos tentariam distrair a pessoa, e falhar significava sacrificar a própria vida.

A tarefa é determinar, dado dois círculos (um desenhado por um caçador e outro representando a área da flor), se o caçador morre ou fica rico com sua conquista.

## **Entrada**

A entrada é composta por diversas instâncias e termina com final de arquivo (**EOF**). Cada instância consiste em uma linha com seis inteiros, R1 (1 ≤ R1) , X1(|X1|), Y1(|Y1|), R2 (R2 ≤ 1000), X2(|X2|), Y2 (Y2 ≤ 1000). O círculo desenhado pelo caçador possui raio R1 e centro (X1; Y1). O círculo representando a área da flor possui raio R2 e centro (X2; Y2).

## **Saída**

Para cada instância imprima uma única linha contendo MORTO, se o caçador morre, ou RICO se o caçador consegue colher a flor.

**from math import sqrt**

**while True:**

**try: R1, X1, Y1, R2, X2, Y2 = [int(x) for x in input().split()]**

**except: break**

**D = sqrt((X1-X2)\*\*2 + (Y1-Y2)\*\*2)**

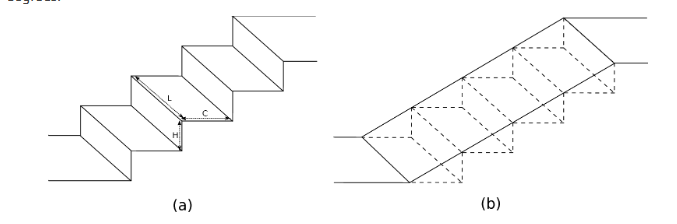
**if R1 >= D+R2: print('RICO')**

**else: print('MORTO')**

**2 - Escada do DINF**

**Contexto**

O prédio do Departamento de Ilhas Naturais Federais (DINF) está passando por uma reforma para deixá-lo mais acessível. No prédio há uma escada de **N** degraus. Cada degrau tem **H** centímetros de altura, **C** centímetros de comprimento e **L** centímetros de largura. A figura (a) exemplifica uma escada com **N**=4 degraus.



Para tornar o prédio mais acessível, o chefe do DINF decidiu colocar uma rampa sobre a escada. A rampa é rígida e tem forma retangular. Ela será colocada sobre a escada de forma a cobrir todos os seus degraus, como indicado pela figura (b).

Sua tarefa é, dado o número de degraus e suas medidas, determinar qual deve ser área total da superfície da rampa.

**Entrada**

A entrada contém vários casos de teste. A primeira linha de cada caso contém um inteiro N (1 ≤ N ≤ 1000), o número de degraus na escada. A segunda linha contém três inteiros H, C e L (1 ≤ H, C, L ≤ 100), as medidas de cada degrau, em centímetros. A entrada termina com fim-de-arquivo (EOF).

**Saída**

Para cada caso de teste, imprima uma linha contendo a área total da superfície da rampa, em metros quadrados. Arredonde e imprima este valor com exatamente 4 casas decimais.

**from math import sqrt**

**while True:**

**try: n = int(input())**

**except: break**

**h, c, l = [int(x) for x in input().split()]**

**a = sqrt(h\*\*2 + c\*\*2) \* n \* l**

**print('{:.4f}'.format(a/10000))**

**3 - Balão++**

**Contexto**

Nós, os autores (Diego Rangel, Francisco Arcos, Gabriel Duarte e Gustavo Policarpo), estamos felizes por você estar tentando resolver nossos problemas. Para você que é iniciante não sair da sala sem nenhum balão, aqui vai um desafio para você:

* Neste ano os balões têm formato esférico, segundo a empresa que produz os balões: "(...) por motivos complexos de engenharia esse formato é melhor (...)" vai entender...
* Entretanto esse formato faz com que o balão use mais gás hélio e isso causou um problema, pois o organizador já havia comprado um tanque com **L** litros de gás antes dessa novidade no mercado de balões.

Sabendo o raio do modelo de balões e a quantidade de gás hélio disponível, você poderia ajudar a equipe dizendo quantos balões podem ser enchidos completamente?

## **Entrada**

A entrada é composta por dois inteiros **R** e **L** (1 ≤ **R**, **L** ≤ 109), que são o raio e a quantidade de gás disponível, respectivamente.

**Considere PI = 3.1415**

## **Saída**

Você deve imprimir um único inteiro representando a quantidade de balões que podem ser enchidos completamente com a quantidade de gás hélio disponível.

Estruturas e bibliotecas

## 1- Distância Euclidiana

**Contexto**

Italone está participando de uma maratona de programação, mas antes da competição ele vai à praia e se perde. Uma pessoa encapuzada lhe dá uma sequência de instruções (cima, baixo, esquerda, direita) para chegar ao local da competição. Italone deve verificar se essas instruções são corretas ou uma armadilha.

### **Objetivo**

Determinar se Italone consegue chegar ao local da competição seguindo as instruções sem exceder uma distância euclidiana máxima k em qualquer ponto do trajeto.

### **Entrada**

1. **Primeira linha**: Quatro inteiros n, k, xf, yf onde:
   * n é o número de instruções (1 ≤ n ≤ 10^4).
   * k é a distância máxima permitida para não ser uma armadilha (0 ≤ k ≤ 10^4).
   * xf e yf são as coordenadas finais do local da prova (0 ≤ xf, yf ≤ 10^4).
2. **Próximas n linhas**: Cada linha contém um caractere C, B, E, D representando as direções (Cima, Baixo, Esquerda, Direita).

### **Saída**

1. Se Italone começa a mais de k de distância, imprima "Trap 1".
2. Se em qualquer ponto durante as instruções ele exceder a distância k, imprima "Trap" seguido pelo número da instrução onde isso ocorre.
3. Se Italone chega ao local da prova sem exceder a distância k, imprima "Sucesso".

from math import sqrt

p = lambda xi, yi, xf, yf, k: sqrt((xi-xf)\*\*2 + (yi-yf)\*\*2) > k

n, k, xf, yf = [int(x) for x in input().split()]

xi = yi = 0

r = 'indo'

if p(0, 0, xf, yf, k): r = 'Trap 1'

for g in range(n):

o = input()

if o == 'C': yi += 1

elif o == 'B': yi -= 1

elif o == 'E': xi -= 1

elif o == 'D': xi += 1

if r == 'indo' and xi == xf and yi == yf: r = 'Sucesso'

if r == 'indo' and (g+1 == n or p(xi, yi, xf, yf, k)):

r = 'Trap ' + str(g+1)

print(r)

2- **Resumo do Problema**

Dâmi e Marcus criaram um jogo que utiliza uma matriz e várias operações para modificá-la e consultá-la. Você deve implementar um programa que gerencie essas operações eficientemente.

### **Operações**

1. **Atualização de Submatriz (U X Y Z W V)**:
   * Incrementa um valor V em todas as posições dentro da submatriz definida pelo canto superior esquerdo (X, Y) e canto inferior direito (Z, W).
2. **Consulta de Valor (A X Y)**:
   * Retorna o valor na posição (X, Y) da matriz.

### **Entrada**

* Um inteiro Q representando o número de operações (1 ≤ Q ≤ 100,000).
* Q operações no formato especificado acima.

### **Saída**

* Para cada operação do tipo "A", imprimir o valor correspondente da matriz.

### **Desafios**

* Eficiência na atualização de grandes submatrizes e na consulta de valores específicos.
* Manter a complexidade computacional baixa para lidar com até 100,000 operações.

m = [[0 for x in range(500)] for y in range(500)]

for p in range(int(input())):

e = input().split()

if e[0] == 'U':

x1 = int(e[1])

y1 = int(e[2])

x2 = int(e[3])

y2 = int(e[4])

v = int(e[5])

for i in range(x1-1, x2):

for j in range(y1-1, y2): m[i][j] += v

else: print(m[int(e[1])-1][int(e[2])-1])

## 3- Somar frações

**Resumo do Problema**

Joãozinho precisa de um programa que some duas frações e forneça o resultado na forma irredutível.

### **Entrada**

* Quatro inteiros a, b, c, d:
  + a/b é a primeira fração.
  + c/d é a segunda fração.
  + Todos os inteiros estão no intervalo de 1 a 100.

### **Saída**

* Dois inteiros representando a fração resultante da soma das duas frações na sua forma irredutível.

### **Etapas para Solução**

1. **Somar as frações**:
   * Encontrar o denominador comum.
   * Calcular o numerador da soma.
2. **Simplificar a fração**:
   * Encontrar o máximo divisor comum (MDC) do numerador e denominador.
   * Dividir numerador e denominador pelo MDC para obter a fração irredutível.

def div(n):

d, m = [], n // 2

d.append(n)

for i in range(m, 1, -1):

if n % i == 0: d.append(i)

return d

a, b, c, d = [int(w) for w in input().split()]

y = b \* d

x = int((y/b\*a) + (y/d\*c))

c = True

while c:

dx, dy = div(x), div(y)

p = 0

for e in dx:

if e in dy:

p = e

break

if p:

x //= p

y //= p

else: c = False

print(x, y)

## 4- Elevador

### **Resumo do Problema**

Você precisa escrever um programa para verificar se a capacidade máxima de um elevador foi excedida em algum momento durante um dia de funcionamento. A entrada fornece leituras de quantas pessoas saem e entram no elevador em cada andar, e o programa deve determinar se, em algum momento, o número de pessoas no elevador ultrapassou a capacidade máxima.

### **Entrada**

1. A primeira linha contém dois inteiros N e C:
   * N: número de leituras realizadas pelo sensor (1 ≤ N ≤ 1000).
   * C: capacidade máxima do elevador (1 ≤ C ≤ 1000).
2. As próximas N linhas contêm dois inteiros S e E cada, onde:
   * S: número de pessoas que saíram no andar.
   * E: número de pessoas que entraram no andar.

### **Saída**

* O programa deve imprimir:
  + S se a capacidade do elevador foi excedida em algum momento.
  + N caso contrário.

n, c = [int(x) for x in input().split()]

t = 0

r = 'N'

for g in range(n):

s, e = [int(x) for x in input().split()]

t = t - s + e

if t > c: r = 'S'

print(r)

## 5- Economia de gasolina

### **Resumo do Problema**

A Companhia de Táxi Tabajara (CTT) precisa determinar se é mais econômico abastecer seus carros bi-combustíveis com álcool ou gasolina. Dada a variação dos preços e o desempenho dos carros com cada combustível, você deve escrever um programa para calcular qual combustível é mais econômico.

### **Entrada**

Uma linha com quatro números reais:

* A: preço por litro do álcool (0.01 ≤ A ≤ 10.00)
* G: preço por litro da gasolina (0.01 ≤ G ≤ 10.00)
* Ra: rendimento (km/l) do carro com álcool (0.01 ≤ Ra ≤ 20.00)
* Rg: rendimento (km/l) do carro com gasolina (0.01 ≤ Rg ≤ 20.00)

### **Saída**

* Imprimir 'A' se é mais econômico abastecer com álcool.
* Imprimir 'G' se é mais econômico ou indiferente abastecer com gasolina.

a, g, ra, rg = [float(x) for x in input().split()]

if ra/a <= rg/g: print('G')

else: print('A')

## 6- Fases da Lua

### **Resumo do Problema**

Jade, com seu telescópio, observa a iluminação da lua em duas noites consecutivas. Com base nesses percentuais de iluminação, você precisa determinar a fase da lua na noite anterior. As fases da lua são determinadas conforme os seguintes intervalos:

* **Lua Nova**: 0% a 2% de iluminação.
* **Lua Crescente**: 3% a 96% de iluminação (e crescente em relação à noite anterior).
* **Lua Cheia**: 97% a 100% de iluminação.
* **Lua Minguante**: 96% a 3% de iluminação (e decrescente em relação à noite anterior).

### **Entrada**

* Uma linha com dois inteiros:
  + Primeiro valor: percentual de iluminação de dois dias atrás.
  + Segundo valor: percentual de iluminação da noite anterior.

### **Saída**

* Imprimir a fase da lua da noite anterior:
  + "nova" para Lua Nova.
  + "cheia" para Lua Cheia.
  + "crescente" para Lua Crescente.
  + "minguante" para Lua Minguante.

e = str(input()).split()

a = int(e[0])

b = int(e[1])

if a < b:

if b <= 2: print('nova')

elif b <= 96: print('crescente')

else: print('cheia')

else:

if b >= 97: print('cheia')

elif b >= 3: print('minguante')

else: print('nova')

## 7- CPF

### **Resumo do Problema**

O problema consiste em determinar se um CPF é válido ou não. Um CPF é considerado válido se a soma ponderada dos seus dígitos resultar em um número múltiplo de 11. Para isso, são utilizados os dígitos verificadores b1 e b2, que são calculados de acordo com regras específicas.

### **Entrada**

* Um número desconhecido de CPFs, não excedendo 10000 casos.
* Cada linha contém um CPF no formato:
  + d1d2d3.d4d5d6.d7d8d9-d10d11

### **Saída**

* Para cada CPF informado:
  + Se o CPF for válido, imprimir "CPF valido".
  + Se o CPF for inválido, imprimir "CPF invalido".

### **Abordagem**

1. Iterar sobre cada CPF na entrada.
2. Para cada CPF, calcular os dígitos verificadores b1 e b2 de acordo com as regras fornecidas.
3. Verificar se a soma ponderada dos dígitos resulta em um número múltiplo de 11.
4. Imprimir "CPF valido" se for válido ou "CPF invalido" se for inválido.

while True:

try:

e = str(input()).strip()

c = e.replace('.', '').replace('-', '')

s = 0

for i in range(9):

s += int(c[i]) \* (i + 1)

b1 = s % 11

b1 = 0 if b1 == 10 else b1

s = 0

if b1 == int(e[-2]):

for i in range(9):

s += int(c[i]) \* (9 - i)

b2 = s % 11

b2 = 0 if b2 == 10 else b2

if b2 == int(e[-1]):

print('CPF valido')

else: print('CPF invalido')

else: print('CPF invalido')

except EOFError: break

## 8- Matriz de quadrados

O problema solicita a implementação de um programa que calcule o "quadrado" de cada matriz fornecida. Para Atrapalhilton, o "quadrado" de uma matriz é obtido elevando ao quadrado cada elemento da matriz original. A entrada consiste em um inteiro N, seguido pela descrição de N matrizes. Cada matriz é representada por um número M (o número de linhas e colunas) e M linhas contendo os valores das células da matriz. A saída deve mostrar o "quadrado" de cada matriz, alinhando os valores de cada coluna à direita e deixando uma linha em branco entre os "quadrados" das matrizes consecutivas.

def imp(n, e):

print(' ' \* (e - len(str(n))), end='')

print(n, end='')

return

c = 4

for g in range(int(input())):

n = int(input())

v = []

for i in range(n):

t = [int(x)\*\*2 for x in str(input()).split()]

v.append(t)

if c != 4: print()

print('Quadrado da matriz #{}:'.format(c))

c += 1

esp = []

for i in range(n):

m = 0

for j in range(n):

if len(str(v[j][i])) > m: m = len(str(v[j][i]))

esp.append(m)

for i in range(n):

for j in range(n):

imp(v[i][j], esp[j])

if j != n-1: print(end=' ')

print()

## 9- Pares de bota pode ser formado

O problema solicita a implementação de um programa que, dada uma lista contendo a descrição de cada bota entregue, determine quantos pares corretos de botas podem ser formados no total.

### **Entrada**

* A entrada consiste em vários casos de teste.
* Cada caso de teste começa com um inteiro NNN (2 ≤ N≤104N ≤ 10^4N≤104), indicando o número de botas individuais entregues.
* Em seguida, há NNN linhas descrevendo cada bota, contendo um número inteiro MMM (30 ≤ M≤60M ≤ 60M≤60) representando o tamanho da bota e uma letra LLL indicando o pé da bota: 'D' para o pé direito e 'E' para o pé esquerdo.

### **Saída**

* Para cada caso de teste, imprimir uma linha contendo um único número inteiro indicando o número total de pares corretos que podem ser formados.

Este problema pode ser resolvido contando o número de botas de cada tamanho e para cada tamanho, dividindo esse número por 2 e somando os resultados para obter o total de pares corretos de botas

while True:

try:

n = int(input())

v = []

q = 0

for i in range(n):

t = []

p = []

e = str(input()).split()

t.append(int(e[0]))

p.append(int(e[0]))

t.append(e[1])

if e[1] == 'E': p.append('D')

else: p.append('E')

if p in v:

v.remove(p)

q += 1

else: v.append(t)

print(q)

except EOFError: break