1. Enumere os componentes da arquitetura de von Neumann.

2. Um estudante deseja escrever um programa para calcular sua média final em uma disciplina, depois de ter feito a prova final. Ele obteve média parcial 6.0 (seis) e, na prova final, obteve nota 5.0 (cinco). A média final, na UFCG, é calculada como a média ponderada da média parcial e da nota na prova final, com pesos 60% e 40%, respectivamente. Pede-se, escreva a expressão em python que permite calcular a média final do referido aluno.

3. Um estudante criou o pequeno trecho de programa abaixo para calcular a média de suas notas em uma disciplina de programação. O estudante, contudo, não consegue entender por que o programa não funciona corretamente. Ao testá-lo com as notas registradas no código do programa, por exemplo, a média impressa não é a que ele espera (7.5). Pede-se que você analise a expressão usada no programa, determine a razão pela qual ele não funciona como esperado e o modifique, fazendo-o funcionar corretamente. Envie como resposta o programa modificado.

```
# coding: utf-8
# Cálculo da média em Álgebra Vetorial

nota1 = 10
nota2 = 5

media = nota1 + nota2 / 2

print "Nota 1: %4.1f" % nota1
print "Nota 2: %4.1f" % nota2
print "Média : %4.1f" % media
```

4. O programa incompleto abaixo deve calcular a área de um círculo cujo raio será lido da entrada. A área corresponde ao produto de pi (math.pi em python) pelo quadrado do raio. Pede-se que você complemente o programa abaixo, escrevendo a expressão correta para a área e complementando o programa com o que for necessário para que funcione corretamente.

```
# coding: utf-8
# Cálculo da área de um círculo
# Dalton Serey, 2012

raio = float(raw_input("Qual o raio do círculo?"))

area =

print "Raio: %4.1f" % raio
print "Área: %4.1f" % area
```

5. Defina o que é um programa.

6. Escreva um programa que lê seu nome da entrada e imprime a mensagem "Oi Fulano", em que "Fulano" é substituído pelo nome digitado. Veja o exemplo de execução abaixo.

Entrada	Saída
Dalton	Oi Dalton

7. Escreva um programa que imprime a soma dos 10 primeiros números naturais positivos. As operações de soma devem ser realizadas pelo computador, não por você. Apenas o número resultante deve ser impresso.

8. Escreva um programa que imprime o produto dos números primos menores que 20. Apenas o número resultante deve ser impresso.

9. Ecreva um programa que leia um número da entrada e imprima o seu dobro. Veja o exemplo de execução abaixo.

Entrada	Saída
5	10

10. Escreva um programa que imprime a média ponderada entre 5.0, 8.0 e 10.0, com os pesos 4.0, 3.5 e 2.5, respectivamente. Apenas o número deve ser impresso.

11. Escreva um programa que leia um número de ponto flutuante e que calcule e imprima apenas sua parte fracionária (o número menos sua parte inteira). Por exemplo, se o número 1.5 for dado na entrada, o programa deve imprimir 0.5 na saída. Veja o exemplo de execução abaixo.

Entrada	Saída
1.3	0.3

12. Escreva um programa que leia seu nome da entrada e o imprima 3 vezes. Veja o exemplo de execução abaixo (observe que está em azul o que foi digitado pelo usuário na entrada do programa).

Entrada	Saída
Dalton	Dalton Dalton Dalton

13. O movimento retilíneo uniforme de um corpo qualquer é modelado pela seguinte equação: s(t) = s0 + vt, em que s(t) indica a posição do corpo no instante t, s0 indica a posição incial do corpo e v sua velocidade. Pede-se: escreva um programa que leia da entrada os valores de s0, v e t e que imprima na saída a posição do corpo no instante t. Veja o exemplo de entrada e saída abaixo em que o usuário deseja calcular qual a distância percorrida a partir do kilômetro s0 = 0.0, a uma velocidade v = 120.0 km/h, se o corpo se deslocar por t = 4.5h. Observe que o valor de saída deve ser impresso com duas casas decimais.

Entrada	Saída
0.0 120.0 4.5	540.00

14. Escreva um programa que ajude um pintor a calcular o custo de seu serviço. Ele cobra por R$20,00 por metro quadrado de área pintada. O programa deve ler da entrada as dimensões de uma parede retangular (altura, largura) e informar o custo em reais do serviço. Os espaços das portas, janelas ou outros obstáculos nas paredes não são considerados para fins de desconto. Veja os exemplos de entrada e saída.

Entrada	Saída
2.5 5.0	R$ 250.00

15. Escreva um programa que lê três notas parciais e calcula e imprime a média ponderada final de um aluno. Os pesos de cada nota são, respectivamente, 2.0, 3.0 e 5.0. Veja o exemplo de execução abaixo.

Entrada	Saída
10.0 8.0 6.0	7.4

16. Escreva um programa que leia uma temperatura Celsius e imprima a correspondente em Fahrenheit. Veja o exemplo de execução abaixo.

Entrada	Saída
-10	14.0

17. Escreva um programa que receba um número natural que representa o raio de um círculo e calcule a sua área. Dica: Importe o módulo math e use math.pi.

Entrada	Saída
2	12.5663706144

18. Um aluno do curso de elétrica da UFCG pediu aos colegas do curso de computação, um programa para calcular o quanto ele precisaria tirar numa prova final caso tenha obtido uma média parcial maior ou igual a 4.0 e menor do que 7.0 em uma disciplina. Faça um programa que receba uma nota neste intervalo como entrada e calcule a nota minimamente necessária para a prova final para que o aluno seja aprovado. Lembrando que o aluno é aprovado se sua média final for pelo menos igual a 5. Esta média final é calculada pela média ponderada da média parcial (com peso 6) e da prova final (peso 4). Veja exemplos de execução abaixo.

Entrada	Saída
4.0	6.5

19. Em alguns programas de computador, é comum referenciar a hora atual de um sistema através do chamado "Unix time" ou "POSIX time". Este valor é calculado pelo número de segundos que se passaram desde a hora zero do dia 1º de Janeiro de 1970. Crie um programa que recebe esta quantidade de segundos e que tenha como saída a quantidade de dias completos que este valor representa. Veja o exemplo de execução abaixo.

Entrada	Saída
1330319498	15397
7126312312	82480

20. Escreva um programa que ajude um pintor a calcular quantos litros de tinta são necessários para pintar uma parede. Segundo o fabricante, um galão de tinta (3.6 l) é suficiente para pintar 50 m2. O programa deve ler da entrada as dimensões de uma parede retangular (altura, largura) e informar, na saída, a quantidade de litros de tinta com duas casas decimais. Os espaços das portas, janelas ou outros obstáculos nas paredes não são considerados para fins de desconto. Veja os exemplos de entrada e saída.

Entrada	Saída
2.5 5.0	0.90 l

21. Você deve escrever um programa que determina o número verificador de números de conta corrente de um banco. O número de conta corrente possui exatamente 5 dígitos. Para obter o número verificador de uma conta é necessário somar todos os algarismo que compõe o número. O resto da divisão deste somatório pelo número 11 é o número verificador. Seu programa deve ler da entrada um número de 5 dígitos que representa o número de uma conta corrente. Na saída, deve escrever o número da conta e seu número verificador, no seguinte formato -. O número verificador deve ser mostrado com dois dígitos. Observe os exemplos de entrada e saída.

Entrada	Saída
01234	01234-10
40016	40016-00

22. Um professor de matemática precisa corrigir rapidamente as soluções de seus alunos para equações de segundo grau. Para ajudá-lo, você deve escrever o programa eq2grau.py que calcula as raízes reais de uma equação de segundo grau. O programa lê da entrada os três coeficientes (a, b e c) e imprime na saída as raízes, se existirem. A saída deve seguir os exemplos abaixo.

Entrada	Saída
3 -7 2	x1 = 2.00 x2 = 0.33

5 -6 5	sem raizes reais
-1 4 -4	x = 2.00

23. Escreva um programa que leia da entrada, inicialmente, o número de gols marcados pelo Campinense e, em seguida, o número de gols marcados pelo Treze num "Clássico dos Maiorais" e que imprima o nome do time vencedor ou a palavra “Empate”, caso o clássico termine empatado.

Entrada	Saída
2 1	Campinense
2 2	Empate
2 4	Treze

24. Escreva um programa que leia da entrada um valor inteiro e verifica se o valor recebido representa um ano bissexto ou não. O resultado da verificação deve ser exibido na saída do programa da seguinte forma: " não é bissexto" ou " é bissexto". Veja exemplo de entrada/saída abaixo. Dica: um ano é bissexto se for divisível por 400 ou se for divisível por 4 e não por 100.

Entrada	Saída
2011	2011 não é bissexto
2012	2012 é bissexto

25. Escreva um programa que receba um número de 5 dígitos da entrada e calcule o produto entre: o somatório dos dígitos que encontram-se em posições pares e o somatório dos dígitos que encontram-se em posições ímpares. O valor resultante deve ser mostrado na saída, com exatamente cinco dígitos. Complete com 0 (zeros) caso necessário. Veja exemplos de entrada e saída.

Entrada	Saída
12344	00048
12221	00016

26. Escreva um programa que receba um número de 5 dígitos da entrada e calcule a soma do número formado pelos dois primeiros dígitos com o número "formado pelos dois últimos dígitos. O valor resultante deve ser multiplicado pelo dígito do meio e isto deve ser mostrado na saída, com exatamente quatro dígitos. Complete com 0 (zeros) caso necessário. Veja exemplos de entrada e saída.

Entrada	Saída
12145	0057
10220	0060

27. Os números de matrículas utilizados na UFCG passaram por uma mudança em 2012. Um exemplo típico de número de matrícula de um aluno do curso de Ciência da Computação era 20111035 (sempre iniciando com o número

dois e possuindo oito dígitos no total). Com a nova alteração as matrículas passaram a possuir nove dígitos e para fazer a conversão para o novo formato de matrícula basta seguir dois passos: 1) inserir um zero após o quinto dígito da matrícula antiga; 2) trocar o dígito inicial por 1. Escreva um programa que realiza tal conversão. Veja um exemplo de entrada e saída abaixo.

Entrada	Saída
20511035	105110035

28. O DNA humano é formado por quatro tipos de nucleotídeos: adenina (A), guanina(G), citosina(C) e timina (T). Eles são unidos para formar sequências. Escreva um programa que receba três seqüências de DNA e informe, na saída qual é a menor sequência e o seu tamanho. Caso o menor tamanho seja o mesmo para duas ou três sequências, considere a que foi informada primeiro. Veja exemplos de entrada e saída.

Entrada	Saída
AGCT TTTGGA ACT	ACT 3
CCGG CCAA ACTGGG	CCGG 4

29. Escreva um programa que receba três números inteiros da entrada, calcule e imprima a soma deles. Entretanto, se um número 13 for informado, ele não deve ser incluído na soma, bem como, nem um outro número depois dele. Se a soma dos números for 13, um zero deve ser mostrado na saída. Veja exemplos de entrada e saída.

Entrada	Saída
8 5 13	0
1 2 3	6
13 1 2	0
1 2 13	3

30. Seu João é motorista de transporte alternativo e faz a linha Campina Grande-João Pessoa todos os dias. Ele deseja calcular o rendimento de seu carro e também o lucro obtido em cada dia de trabalho. Para ajudar seu João, você deve escrever um programa que recebe como entrada: a marcação do hodômetro (Km) no início do dia, a marcação (Km) no final do dia, o número de litros de combustível gasto e o valor total (R$) recebido dos passageiros. Considerando que um litro de combustível custa R$ 2,55, seu programa deve calcular e imprimir a média do consumo em Km/L e o lucro (líquido) do dia. Observe os exemplos abaixo para entender como o programa deve funcionar.

Entrada	Saída
50120 50452 37 170.00	Consumo em Km/L: 8 Lucro liquido: 75.65
10200	Consumo em Km/L: 9

10500 32 150.00	Lucro liquido: 68.40

31. Escreva um programa que lê da entrada as dimensões de uma cozinha retangular (comprimento, largura e altura), calcula e escreve a quantidade de caixas de azulejos necessárias para cobrir todas as suas paredes. Não precisa levar em consideração espaços para colocar portas e janelas. Considere que cada caixa de azulejos possui 1,5 m2. Veja abaixo exemplos de execução do programa.

Entrada	Saída
3.5 2 2.5	Quantidade de caixas: 19
5 4 3	Quantidade de caixas: 36

32. Escreva um programa que calcula o peso ideal com base no sexo e na altura de uma pessoa. Para homens, o peso é calculado através da fórmula "72.7 * altura - 58". Para mulheres, a fórmula é "62.1 * altura - 44.7". O programa deve ler, da primeira linha da entrada, um caractere que indica o sexo da pessoa ("m" para masculino e "f" para feminino) e, em seguida, deve ler o valor que representa a altura da pessoa. Em seguida, o programa deve imprimir o valor do peso ideal de acordo com as fórmulas indicadas. Observe que o valor de saída deve ser impresso com três casas decimais. Veja exemplos de execução abaixo.

Entrada	Saída
m 1.80	72.860
f 1.50	48.450

33. Escreva um programa que receba dois números inteiros da entrada. O programa deve imprimir na saída SIM se: Um dos números é o dobro do outro. Caso contrário o programa imprime NAO.Veja exemplos de execução abaixo.

Entrada	Saída
5 10	SIM
3 3	NAO

34. O status final de um estudante da UFCG depende de suas notas nos estágios e de seu número de faltas. Ele é considerado aprovado por média se tiver pelo menos 75% de presença e se sua média for maior ou igual a 7,0 (sete). É considerado reprovado por faltas se tiver menos de 75% de presença (reprovado por faltas). Se tiver presença suficiente, mas se sua média for inferior a 4,0 (quatro) é considerado reprovado por nota. Se tiver presença suficiente, mas se sua média for entre 4,0 e 7,0, o aluno deve ir pra prova final. Pede-se: escreva um programa que leia as três notas do aluno e seu número (percentual) de faltas e que imprima na saída o status do aluno, conforme a lógica indicada. Observe os exemplos de entrada e saída.

Entrada	Saída
4.0 6.0 5.0 25	reprovado por faltas

4.0 2.0 3.0 80	reprovado por nota
10.0 10.0 10.0 70	reprovado por faltas
4.0 6.0 5.0 85	prova final
10.0 10.0 10.0 78	aprovado por media

35. Escreva um programa que imprima todos os números da sequência

```
8.8, 9.0, 9.2, 9.4, ..., 100.0
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

36. Escreva um programa que imprima todos os números da sequência

```
15.2, 13.7, 12.2, ..., -4.3, -5.8
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

37. Escreva um programa que imprima todos os números da sequência

```
302, 299, 296, ..., -7, -10
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

38. Escreva um programa que imprima todos os números da sequência

```
4.0, 4.5, 5.0, …, 103.0, 103.5
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

39. Escreva um programa que imprima todos os números da sequência

```
4, 7, 10, ..., 2998, 3001
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

40. Escreva um programa que imprima todos os números da sequência

```
1, 5, 9, ..., 393, 397
```

O programa deve imprimir cada número em uma linha separada da saída. Nada mais, além dos números, deve ser impresso.

41. Escreva um programa que recebe três números na entrada e calcula o somatório deles. O programa deve imprimir "plif" se este somatório for divisível por 3 ou "plof" se for divisível por 5. Se for divisível por ambos, imprima "plifplof". Caso não seja divisível por nenhum, imprima o somatório. Veja exemplo de entrada e saída.

Entrada	Saída
5 2 3	plof

4 4 4	plif

42. Escreva um programa que conta quantas vogais existe em uma palavra lida na primeira linha da entrada. A saída do programa deve explicitar a quantidade de cada uma das vogais. O programa não recebe palavras acentuadas e não deve diferenciar letras maiúsculas de minúsculas.

Entrada	Saída
Anaconda	A - 3 E - 0 I - 0 O - 1 U - 0

43. Para que se possa construir um triângulo é necessário que a medida de qualquer um dos lados seja menor que a soma das medidas dos outros dois e maior que o valor absoluto da diferença entre essas medidas.
Escreva um programa receba três medidas e verifica se elas obedecem à condição de existência do triângulo formando um triângulo válido. Em caso afirmativo, imprima "triangulo valido." e o seu perímetro. Caso contrário, imprima "triangulo invalido.". Veja exemplos de entrada e saída.

Entrada	Saída
2 2 2	triangulo valido. 6
20 10 5	triangulo invalido.

44. Escreva um programa que imprima a sequência de números ímpares entre 1 e 101 (ambos incluídos), mas com os números divisíveis por 3 ou por 5 substituídos por um asterisco (“*”). Veja a sequência abaixo. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.

```
1, *, *,  7, *, 11, 13, ..., 101
```

45. Escreva um programa que leia da entrada três valores inteiros, verifica se os mesmos correspondem aos lados de um triângulo válido e imprime o tipo do triângulo (Escaleno, Isósceles ou Equilátero). Caso os valores recebidos não correspondam a um triângulo válido, a seguinte mensagem deve ser retornada: "Não é triângulo".

Entrada	Saída
2 2 2	Equilátero
20 10 5	Não é triângulo

46. Escreva um programa que leia dois números inteiros X e Y e imprima uma tabela dos quadrados dos valores entre X e Y, inclusive, e que indique, linha a linha, quais dos quadrados são divisíveis por 3, com um asterisco ("*"). Se X > Y, o programa deve imprimir apenas uma linha contendo "---". Veja os exemplos de entrada e saída.

Entrada	Saída
15 19	15 225 * 16 256

	17 289 18 324 * 19 361
11 7	---
22 22	22 484

47. Escreva um programa que leia dois números inteiros X e Y e imprima uma tabela dos quadrados dos valores entre X e Y, inclusive. Se X > Y, o programa deve imprimir apenas uma linha contendo "---". Veja os exemplos de entrada e saída.

Entrada	Saída
15 19	15 225 16 256 17 289 18 324 19 361
11 7	---
22 22	22 484

48. O programa abaixo lê uma sequência de N temperaturas observadas em graus celsius e as coloca em uma lista (N é o primeiro valor lido da entrada). Pede-se que você modifique o programa e o faça imprimir um relatório contendo uma tabela das diferenças entre cada temperatura e a temperatura média observada em todo o período. Observe o exemplo de entrada e saída e siga a formatação de dados.

Entrada	Saída
4 18.9 19.5 20.1 20.0	18.90 -0.73 19.50 -0.12 20.10 0.48 20.00 0.38
4 19.2 19.6 19.8 19.7	19.20 -0.38 19.60 0.03 19.80 0.23 19.70 0.12

49. Escreva um programa que imprima a sequência de números ímpares entre 1 e 101 (ambos incluídos), mas com os números divisíveis por 7 ou terminados em 7 substituídos por um asterisco (“*”). Veja a sequência abaixo. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.

```
1, 3, 5, *, 9, 11, 13, 15, *, 19, *, 23, ..., 101
```

50. Escreva um programa que imprima a sequência de números ímpares entre 1 e 101 (ambos incluídos), mas com os números divisíveis por 5 mas não divisíveis por 3 substituídos por um asterisco (“*”). Veja a sequência abaixo. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.

```
1, 3, *, 7, 9, 11, 13, 15, ..., 101
```

51. Escreva um programa que imprima a sequência de números ímpares entre 1 e 101 (ambos incluídos), mas com os números divisíveis por 3 substituídos por um asterisco ("*"). Veja a sequência abaixo. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.

```
1, *, 5, 7, *, ..., 97, *, 101
```

52. Escreva um programa que imprima a sequência de números ímpares entre 1 e 101 (ambos incluídos), mas com o número 73 substituído por um asterisco ("*"). Veja a sequência abaixo. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.

```
1, 3, ..., 69, 71, *, 75, 77, ..., 101
```

53. Escreva um programa que leia três números inteiros a, r e n da entrada e que imprima os n primeiro termos de uma progressão aritmética (p.a.). Considere que a é o primeiro termo, r é a razão da p.a. e n é a quantidade de termos a serem impressos. Veja o exemplo de entrada e saída abaixo.

Entrada	Saída
10 2 3	10 12 14

54. Escreva um programa que calcula o percentual de reajuste do salário mínimo. O programa deve ler da entrada o valor atual do salário mínimo e o novo valor, calcular o percentual de reajuste e imprimir o resultado na saída. Observe os exemplos de execução abaixo, para saber como formatar entrada e saída.

Entrada	Saída
500.00 560.00	Valor atual? Novo valor? Reajuste de 12.0%
500.00 545.00	Valor atual? Novo valor? Reajuste de 9.0%

55. Uma empresa mantém em duas listas os valores totais de receitas e de despesas mensais. Escreva um programa que leia as duas listas e que imprima a lista dos lucros de cada mês em que houve prejuízo. O programa deve ler de cada linha da entrada um par de valores de receita e de despesa de cada vez, separados por espaço. A unidade dos valores não é informada, mas é a mesma em todos eles. O programa deve calcular e imprimir, mês a mês, o lucro da empresa. Os lucros devem ser impressos com uma casa decimal de precisão e de acordo com a formatação abaixo.

Entrada	Saída
5.1 4.1 7.4 5.4 8.2 2.7 7.5 8.8 6.4 5.4 9.5 6.5 8.4 2.2 1.4 4.1 5.2 4.7 1.0 2.1 9.8 9.1 5.8 9.8	abr -1.3 ago -2.7 out -1.1 dez -4.0

56. Uma empresa mantém em duas listas os valores totais de receitas e de despesas mensais. Escreva um programa que leia as duas listas e que imprima a lista dos lucros de cada mês em que houve prejuízo. O programa deve ler as receitas e as despesas das duas primeiras linhas da entrada, respectivamente. Os dados de cada mês estão separados um do outro por espaços. A unidade dos valores não é informada, mas é a mesma em todos eles. O programa deve calcular e imprimir, mês a mês, o lucro da empresa. Os lucros devem ser impressos com uma casa decimal de precisão e de acordo com a formatação abaixo.

Entrada	Saída

5.1 7.4 8.2 7.5 6.4 9.5 8.4 1.4 5.2 1.0 9.8 5.8 4.1 5.4 2.7 8.8 5.4 6.5 2.2 4.1 4.7 2.1 9.1 9.8	abr -1.3 ago -2.7 out -1.1 dez -4.0

57. Escreva o programa que receba da entrada um número inteiro maior que zero e imprima na saída os seus divisores próprios pares. Cada divisor deve ser impresso em uma linha separada e nada mais deve ser impresso. Caso o número não tenha divisores pares, nada deve ser impresso. Veja um exemplo de execução abaixo:

Entrada	Saída
12	2 4 6
15	

58. Escreva um programa que desenha uma arvore de natal em forma de um triangulo de asteriscos e um asterisco para representar o tronco. O programa recebe da entrada um numero natural diferente de zero que representa a altura do triangulo Nao deve haver espaco apos o ultimo asterisco da linha. Veja o exemplo de saida:

Entrada	Saída

5	<pre> *
	   ***
	  *****
	 *******
	*********
	    *</pre>

59. Escreva um programa que imprime todos os inteiros positivos divisíveis por A e B ao mesmo tempo, menores ou iguais a K. O programa recebe A, B e K na entrada. Os números devem ser impressos na saída um em cada linha em ordem crescente.

Entrada	Saída
2 4 12	4 8 12

60. Escreva um programa que receba da entrada uma sequência de pelo menos dois números inteiros não negativos separados por espaços. O programa deve imprimir a soma dos dois maiores números desta sequência. Não devem ser usados métodos da classe list.

Entrada	Saída
5 3 4 7 10	17
3 7	10

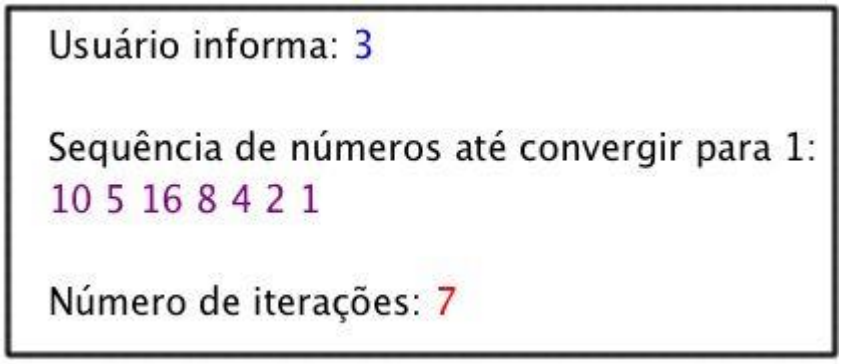
61. Você deve escrever um programa que deve ler da entrada um número binário de 5 bits e efetua a conversão deste número para decimal. Na saída, deve escrever os dois números no seguinte formato: " na base 2 = na base 10". O número decimal deve ser mostrado com dois dígitos. Observe os exemplos de entrada e saída. Não usar nenhuma função de python de conversão de números.

Entrada	Saída
01100	01100 na base 2 = 12 na base 10
00101	00101 na base 2 = 05 na base 10

62. Considere uma sequência de números iniciada em um número natural qualquer e em que cada um dos termos seguintes é determinado em função do anterior, pela seguinte regra:

1) se o número anterior, Ni, for par, o número seguinte será Ni / 2;

2) se o numero anterior, Ni, for ímpar, o número seguinte será 3 * Ni + 1. O matemático Lothar Collatz, em 1937, propôs que todas as sequências formadas através dessa regra, convergem para 1 (por isso, ficou conhecida por conjectura de Collatz). Pede-se que você escreva um programa que receba um número natural positivo e informe quantas iterações são necessárias para convergir para o número 1. Veja o quadro explicativo e exemplo de Entrada e Saída do programa.




```
Entrada
3
Saída
7
```

Entrada	Saída
3	7

63. Escreva um programa que leia um número indeterminado de valores inteiros encerrando a entrada de dados apenas quando um valor menor que zero for informado. O último valor lido é utilizado apenas como critério de parada e deve ser descartado. Após esta entrada de dados, o programa deve imprimir a quantidade de valores lidos da entrada, a soma dos valores e a média, com uma casa decimal. Veja o exemplo de execução abaixo.

```
$ python ex40.py
valor? 10
valor? 8
valor? 7
valor? 6
valor? 9
valor? -1
---
Quantidade de valores: 5
Soma dos valores: 40.0
Média: 8.0
```

Exemplo de entrada e saída no teste do TST:

Entrada	Saída
10 8 7 6 9	valor? valor? valor? valor? valor? valor? --- Quantidade de valores: 5 Soma dos valores: 40.0 Média: 8.0

-1	

64. Escreva um programa que recebe uma sequência de palavras até ler "***". O programa deve imprimir o total de palavras começadas por consoante.

Entrada	Saída
asa controle janela ***	Palavras: 2

65. Escreva um programa que receba palavras até que encontre uma que inicia com a letra 'A'. O programa deve imprimir na saída o numero de palavras lidas antes de encontra-la, em seguida, a palavra iniciada por 'A'. Veja exemplos de saída.

Entrada	Saída
Brasil Equador Chile Argentina	--- 3 Argentina
arquitetura computador Argentina	--- 2 Argentina

66. Escreva um programa que lê palavras da entrada e que, ao final, imprime o comprimento médio dessas palavras (assuma palavras sem acentos e/ou caracteres especiais). O programa deve parar ao ler a palavra “fim” que é apenas um sentinela e não precisa ser considerada para calcular a média de comprimento das palavras. Se nenhuma palavra diferente de "fim" for lida, a média de comprimento deve ser reportada como 0.0. Você pode assumir que nenhuma palavra da entrada terá caracteres especiais (acentos, cedilha, etc). Veja o exemplo de execução.

Entrada	Saída
casa teclados fim	6.0
lua luz tv fim	2.7

67. Escreva um programa para ler vários pares de números, na forma N e K. A cada par N, K lido, calcule e mostre o valor do produto N*K. A leitura deve ser encerrada quando os valores de N e K forem negativos. Veja exemplos de entrada e saída.

Entrada	Saída
2.5 4 3 2.0 -1 -1	10.0 6.0

68. Escreva um programa que leia valores inteiros da entrada e que calcule a média desses valores. O programa deve ler números até que o número 9999 seja lido. O valor 9999 é apenas um “sentinela” para indicar o fim da entrada e não deve ser levado em consideração no cálculo da média. O programa deve imprimir a média com precisão de uma casa decimal. Veja exemplos de entrada e saída.

Entrada	Saída
4 8	6.0

9999	

69. Escreva um programa que leia palavras da entrada até encontrar alguma com número de consoantes maior que o de vogais (assuma palavras sem acentos e/ou caracteres especiais). Na saída deve imprimir o total de palavras lidas. Veja o exemplo de entrada e saída.

Entrada	Saída
mouse casa exemplo	3

70. Escreva um programa que leia pares de números X e Y e que encontre o primeiro par em que X = 2**y (assuma que pelo menos um par será dado). Veja o exemplo de execução. O programa deve imprimir os valores de X e Y do primeiro par que atende à relação.

Entrada	Saída
16 5 1024 9 256 8	256 8

71. Herão de Alexandria foi um matemático de destaque na época em que viveu: entre 150 a.C. a 250 d.C. Em seu trabalho, encontra-se o método de Herão para aproximar a raiz quadrada de um número natural. Para calcular a raiz, partimos de uma aproximação inicial (um chute) e fazemos sucessivos refinamentos até chegar à aproximação da raiz quadrada desejada, respeitando uma margem de erro, ou precisão, previamente definida. Suponha que o número que deseja-se calcular a raiz quadrada é n. Assumindo a0 como uma aproximação inicial, temos a0 e a1:

$$a_0 = n/2$$

$$a_1 = \frac{a_0 + \frac{n}{a_0}}{2}$$

Assim sucessivamente, de modo que a cada iteração melhora-se a aproximação da raiz. Em cada iteração, k assume os valores: k = 1, 2, 3,.... A aproximação dada pela próxima iteração leva em consideração a aproximação da iteração anterior. A regra geral é:

$$a_k = \frac{a_{k-1} + \frac{n}{a_{k-1}}}{2}$$

O critério de parada para o cálculo das novas aproximações considera a precisão desejada para a aproximação da raiz. Se o valor absoluto do quadrado da raiz aproximada ak, subtraída de n for menor que a precisão ε, então tome ak como raiz aproximada. Ou seja, um novo valor de ak deve ser calculado enquanto o erro for menor que a precisão. O erro é dado por:

$E = |(a_k)^2 - n|$. Pede-se que você escreva um programa que receba um número natural e calcule a sua raiz quadrada utilizando o método de Herão. Considere a precisão ε = 10**-4. O programa deve imprimir na saída o valor da aproximação da raiz quadrada formatado com duas casas decimais depois da vírgula e o número de aproximações calculadas até chegar ao resultado final (A aproximação/chute inicial não conta como uma aproximação calculada). Veja exemplos de Entrada e Saída.

```
Entrada
2
Saída
1.41 3


Entrada
4
Saída
2.00 0
```

```
Entrada
25
Saída
5.00 5
```

72. Considere a sequência formada pelos números naturais: 1,2,4,8,16,32,64,128... Escreva um programa que gere e imprima os números da sequência enquanto a soma destes for menor que um limite, um número natural informado pelo usuário. Veja exemplos de Entrada e Saída.

Entrada	Saída
20	1 2 4 8
2	1
10	1 2 4

73. Escreva um programa que lê da entrada um valor do tipo float e imprime o resultado do quadrado desse número em três formas diferentes: sem arredondamento, com arredondamento para cima, com arredondamento para baixo e truncado. Dica: use funções da biblioteca math.

Entrada	Saída
3.8	Valor? Sem arredondamento: 14.44 Com arredondamento para cima: 15.0 Com arredondamento para baixo: 14.0 Valor truncado: 14
2.5	Valor? Sem arredondamento: 6.25 Com arredondamento para cima: 7.0 Com arredondamento para baixo: 6.0 Valor truncado: 6

74. Escreva um programa que lê da entrada o valor final de venda de um automóvel, calcula seu preço sem impostos, os valores pagos para cada tipo de imposto e imprime os resultados. Considere que, para automóveis populares, o ICMS (Imposto sobre Circulação de Mercadorias e Serviços) é de 18%, o IPI (Imposto sobre Produtos Industrializados) é de 13%, o PIS (Programa de Integração Social) é de 1,4% , e a Cofins (Contribuição para o Financiamento da Seguridade Social) é de 7,6%. Todos os impostos são calculados sobre o valor de custo do automóvel. Veja os exemplos abaixo para saber como o programa deve executar.

Entrada	Saída
31500.00	Valor de venda? ICMS: 4050.0 IPI: 2925.0 PIS: 315.0 Cofins: 1710.0 Valor sem impostos: 22500.0
24500.00	Valor de venda? ICMS: 3150.0 IPI: 2275.0 PIS: 245.0 Cofins: 1330.0 Valor sem impostos: 17500.0

75. Escreva um programa que converte N temperaturas Celcius para Fahrenheit. O programa deve ler, da primeira linha da entrada, um inteiro N e, em seguida, deve ler N temperaturas Celsius e imprimir as N temperaturas correspondentes em Fahrenheit. Veja o exemplo de entrada e saída abaixo.

Entrada	Saída

3 -10 -20 0.0	14.0 -4.0 32.0

76. Escreva um programa que leia uma palavra da entrada e que imprima cada letra da palavra, indicando se é vogal ou não.

Entrada	Saída
Prog1	<P> nao <r> nao <o> sim <g> nao <1> nao

77. Um professor precisa calcular rapidamente a nota média de sua turma. Escreva um programa que o ajude. O programa deve receber na primeira linha da entrada o número de alunos da turma N>0, e nas N linhas seguintes, as N notas dos alunos. Na saída, seu programa deve escrever a média dos alunos, arredondada para apenas uma casa decimal. Veja o exemplo de entrada e saída.

Entrada	Saída
3 10.0 8.0 8.0	8.7

78. Um professor precisa calcular rapidamente o número e a média dos alunos que foram reprovados e dos que foram aprovados. Escreva um programa que o ajude. O programa deve receber na primeira linha da entrada o número de alunos da turma N>=1, e nas N linhas seguintes, as N notas dos alunos. Na saída, seu programa deve escrever a média dos alunos reprovados (com nota menor que 7.0), arredondada para apenas uma casa decimal e, na linha seguinte, a média dos alunos aprovados (os com média maior ou igual a 7.0). Se não houver reprovados ou aprovados, o programa não deve apresentar o valor da média. Veja o exemplo de entrada e saída e como deve ser formatada.

Entrada	Saída
1 10.0	Reprovados: 0 Aprovados: 1 Média: 10.0
5 10.0 2.5 8.0 3.0 8.0	Reprovados: 2 Média: 2.8 Aprovados: 3 Média: 8.7

79. Escreva um programa que leia 3 palavras da entrada e imprima a palavra de maior tamanho. Em caso de palavras de tamanhos iguais, imprimir a primeira delas. Você não deve usar funções de ordenação. Veja dois exemplos de entrada e saída abaixo.

Entrada	Saída
bola bicicleta casaco	bicicleta
camisa tv comida	camisa

80. Escreva um programa que leia dois números inteiros, a e b, a <= b, e imprima a soma dos inteiros entre a e b, inclusive. Você não deve usar a função sum(). Veja dois exemplos de entrada e saída abaixo.

Entrada	Saída
5 15	Soma: 110
10 10	Soma: 10

81. Escreva um programa que leia um número inteiro n na primeira linha da entrada e imprima os números ímpares positivos menores que n. Veja exemplos de execução abaixo.

Entrada	Saída
8	1 3 5 7
1	
4	1 3

82. Escreva um programa que receba dois números inteiros da entrada. O programa deve imprimir na saída SIM se: Pelo menos um dos números lidos é 10 ou se a soma dos números é igual a 10. Caso contrário o programa imprime NAO.Veja exemplos de execução abaixo.

Entrada	Saída
10 1	SIM
3 6	NAO

83. Escreva um programa que leia um número inteiro digitado pelo usuário e imprima o seu fatorial. Não use funções nem bibliotecas de Python. Dica: 3! = 3.2.1. Veja exemplos de execução abaixo.

Entrada	Saída
3	6
1	1

84. Escreva um programa que imprima a sequência: 0, 1, 3, 7, 15…, 1023. Apenas um número por linha deve ser impresso.
85. Escreva um programa que imprima todos os números da sequência: 1, 5, 25, 125, … 15625. Cada número deve ser impresso em uma linha separada e nada mais deve ser impresso.
86. Escreva um programa que imprima todos os múltiplos de 5, pares e positivos menores que um dado limite "lim" a ser lido da primeira linha da entrada. Os números devem ser impressos, na saída, um por linha e em ordem crescente.
87. Escreva um programa para ler um valor inteiro n e imprimir os n primeiros elementos da série: 1, 1, 4, 4, 7, 28, 10, 280, 13, …

Exemplo de execução:

```
$ python ex42.py
n? 5
1
1
4
4
7
```

Exemplo de entrada e saída no teste do TST:

Entrada	Saída
5	n? 1 1 4 4 7

88. Escreva um programa que leia da entrada três informações: a) um inteiro n que representa o tamanho da entrada; b) um caractere c que será utilizado na busca; c) uma sequência de n palavras (assuma palavras sem acentos e/ou caracteres especiais). Após a entrada de dados, o programa deve imprimir todas as palavras que começam com o caractere c informado. Veja um exemplo de execução abaixo:

```
$ python ex41.py
n? 3
c? p
palavra? programacao
programacao comeca com p
palavra? Python
Python comeca com p
palavra? LabProg1
LabProg1 nao comeca com p
```

Exemplo de entrada e saída no teste do TST:

Entrada	Saída
3 p programacao Python LabProg1	n? c? palavra? programacao comeca com p palavra? Python comeca com p palavra? LabProg1 nao comeca com p

89. Uma empresa mantém em duas listas os valores totais de receitas e de despesas mensais. Escreva um programa que leia as duas listas e que imprima a lista dos lucros de cada mês. O programa deve ler de cada linha da entrada um par de valores de receita e de despesa de cada vez, separados por espaço. A unidade dos valores não é informada, mas é a mesma em todos eles. O programa deve calcular e imprimir, mês a mês, o lucro da empresa. Os lucros devem ser impressos com uma casa decimal de precisão e de acordo com a formatação abaixo.

Entrada	Saída
5.1 4.1	1.0
7.4 5.4	2.0
8.2 2.7	5.5
7.5 8.8	-1.3
6.4 5.4	1.0
9.5 6.5	3.0
8.4 2.2	6.2
1.4 4.1	-2.7
5.2 4.7	0.5
1.0 2.1	-1.1
9.8 9.1	0.7
5.8 9.8	-4.0

90. Uma empresa mantém em duas listas os valores totais de receitas e de despesas mensais. Escreva um programa que leia as duas listas e que imprima a lista dos lucros de cada mês. O programa deve ler de cada linha da entrada um par de valores de receita e de despesa de cada vez, separados por espaço. A unidade dos valores não é informada, mas é a mesma em todos eles. O programa deve calcular e imprimir, mês a mês, o lucro da empresa. Os lucros devem ser impressos com uma casa decimal de precisão e de acordo com a formatação abaixo.

Entrada	Saída
5.1 4.1 7.4 5.4	jan 1.0 fev 2.0

8.2 2.7	mar 5.5
7.5 8.8	abr -1.3
6.4 5.4	mai 1.0
9.5 6.5	jun 3.0
8.4 2.2	jul 6.2
1.4 4.1	ago -2.7
5.2 4.7	set 0.5
1.0 2.1	out -1.1
9.8 9.1	nov 0.7
5.8 9.8	dez -4.0

91. A carteira nacional de habilitação - CNH é documento obrigatório para todos os motoristas. Se o motorista cometer alguma infração, ganha-se pontos na carteira. A CNH é suspensa caso o motorista acumule 20 pontos ou mais em um ano. As infrações são classificadas de acordo com a sua gravidade, em quatro categorias:

- Gravíssima - 7 pontos
- Grave - 5 pontos
- Média - 4 pontos
- Leve - 3 pontos

Escreva um programa para o DETRAN estadual que receba as infrações que determinado motorista cometeu em um ano, na primeira linha da entrada, e informe a quantidade de pontos totais que ele acumulou na carteira e se sua carteira está suspensa ou não. Veja os exemplos abaixo.

Entrada	Saída
Grave Gravíssima Leve	15 pontos. CNH válida.
Grave Grave Grave Grave Grave	25 pontos. CNH suspensa.

92. Escreva o programa que receba da entrada um número inteiro maior que zero e imprima na saída os seus divisores próprios. Cada divisor deve ser impresso em uma linha separada e nada mais deve ser impresso. Veja um exemplo de execução abaixo:

Entrada	Saída
12	1 2 3 4 6

93. Uma sequência de DNA é uma série de pelo menos quatro letras representando a estrutura primária de uma molécula ou cadeia de DNA. As letras possíveis são A, C, G e T, representando os quatro nucleotídeos existentes. Duas sequencias de DNA de mesmo tamanho são semelhantes se a quantidade de letras iguais, na mesma posição das duas sequencias, é maior do que a metade do tamanho de cada sequencia. Escreva um programa que recebe duas sequencias de DNA e determina se eles são semelhantes ou não. Veja um exemplo de execução abaixo:

Entrada	Saída
ATCGAAGT ATGCTTAT	nao
ATCGAAGT AGCTAAGT	sim

94. Escreva um programa que leia uma palavra da primeira linha da entrada e imprima as letras da palavra lida, alternadamente, começando da primeira letra. Assuma que as palavras são grafadas sem acentos e/ou caracteres especiais. Veja exemplos de execução abaixo.

Entrada	Saída
cabelos	cbls

Ana	Aa

95. Escreva um programa que lê várias palavras da entrada e encontra a primeira vogal de cada palavra. O programa imprime a vogal caso, exista, ou “-”, caso contrário. veja exemplo de entrada e saída. O programa para de ler palavras da entrada ao ler a palavra “fim”, que não deve ser processada.

Entrada	Saída
problema ABC casa exemplo TGV fim	o A a e -

96. Escreva um programa que leia uma palavra da entrada e imprima a primeira vogal desta palavra. Caso não haja vogais na palavra o programa deve imprimir "-". Veja exemplo de execução.

Entrada	Saída
problema	o
CASA	A
CNPQ	-

97. Um meteorologista registra os índices pluviométricos mensais em uma lista de valores. Em outra lista, ele registra os meses correspondentes às medições. Pede-se que você escreva um programa que identifica os meses em que o índice pluviométrico foi abaixo do esperado. O programa recebe na primeira linha da entrada uma lista de índices pluviométricos separados por espaços. Na segunda linha, recebe a indicação dos meses (e ano correspondente) para cada observação (importante: nem todos os meses há observações), separados por espaços, no formato MMM-AA, onde MMM é a abreviatura do mês e AA são os dois dígitos menos significativos do ano. Da terceira linha, será lido apenas o valor mínimo esperado para os índices. A saída do programa consiste em uma linha para cada observação em que é indicado o mês e ano da observação e o valor observado. Observe o exemplo de entrada e saída.

Entrada	Saída
10.1 8.5 20.2 15.5 20.8 13.7 jan-90 fev-90 abr-90 mai-90 jun-90 set-90 15.0	jan-90 10.1 fev-90 8.5 set-90 13.7

98. Você deve escrever um programa que verifica números de conta corrente de um banco. O processo de verificação consiste em checar se o número da conta corresponde ao dígito verificador. Para obter o dígito verificador de uma conta é necessário somar todos os algarismo que compõe o número. O resto da divisão deste somatório pelo número 11 é o dígito verificador. Seu programa deve ler da entrada o par NC, DV que significam, respectivamente, um número de conta e um dígito verificador. Na saída, deve escrever “ok” se a conta estiver válida e “erro” se a conta não atende à restrição. Observe os dados de entrada e saída.

Entrada	Saída
58686 2	erro
51991 3	ok

99. Escreva um programa que receba os nomes e as datas de nascimento de duas pessoas e determina quem das duas é mais velha.

Entrada	Saída
Carmita 24 09 1989 Maroca 12	Carmita

09 1995	
Carol 20 12 2001 Marcos 20 12 2001	nenhuma

100. Os professores de programação 1 da UFCG mantêm em duas listas de mesmo tamanho, o número de exercícios resolvidos por cada aluno e os nomes dos respectivos alunos. Assim, na posição de índice i da lista de exercícios temos o número de exercícios resolvidos pelo aluno da posição de índice i da lista de nomes. Pede-se que você escreva um programa que identifique o aluno que mais resolveu exercícios. O programa deve ler da primeira linha de dados a lista de exercícios resolvidos e da segunda linha os nomes dos alunos. Na saída o programa deve imprimir o nome do aluno que mais resolveu exercícios, bem como o número de exercícios resolvidos. Assuma que não há qualquer possibilidade de empate entre dois alunos.

Entrada	Saída
7 6 8 5 4 8 5 9 4 2 8 a b c d e f g h i j k	h 9

101. Os cartões de crédito com chip (smartcards) requerem de seus usuários uma senha numérica para que a transação financeira seja efetivada. Esta senha é fornecida para os usuários pela administradora de cartões e pode variar de tamanho dependendo da bandeira ou banco do cliente. As senhas obedecem uma lei de formação com o objetivo de torná-las mais seguras. Uma senha segura é composta por um número cuja soma dos algarismos nas posições ímpares resulta em um número ímpar e que a soma dos algarismos nas posições pares resulta em um número par. Por exemplo: Considere a senha segura 7824. A soma do primeiro (7) e do terceiro (2) algarismo (soma = 9) resulta em um número ímpar. A soma do segundo (8) e do quarto (4) algarismo (soma = 12) resulta em um número par.

O seu programa deve receber uma senha numérica e verificar se essa senha é: segura ou insegura. Assuma que a senha tem, pelo menos, dois algarismos. Veja mais exemplos de execução:

Entrada	Saída
125638	segura
1234	insegura

102. Jack é um dos melhores funcionários da transportadora onde ele trabalha. Ele sempre está inovando através da busca de soluções para a redução de custos na empresa. Atualmente, Jack tem observado que um dos pontos fracos da empresa é o setor de carga e descarga de caminhões. Jack está com a ideia de desenvolver um programa que determina o maior tempo gasto para carregar um caminhão e você deve ajudá-lo nesta tarefa.

Entrada

A entrada inicia-se com um inteiro N > 1, indicando o número de carregamentos efetuados no dia. Em seguida, as próximas N linhas são compostas por dois inteiros não negativos que indicam, respectivamente, os horários de início e de conclusão do carregamento.

Saída

Seu programa deve imprimir na saída o carregamento mais demorado. Se houver empate entre dois ou mais carregamentos, então indique apenas o último carregamento entre os empatados.

Observações

Os horários de início e término de um carregamento são sempre exatos e um carregamento sempre inicia e termina num mesmo dia.

Entrada	Saída
3 2 4 14 19 3 6	carregamento 2

103. [Reedição da 61] Escreva o programa que receba da entrada as coordenadas x e y de um ponto do Plano Cartesiano, cada coordenada em uma linha separada. O programa deve imprimir a localização deste ponto com relação aos seguintes locais: “Origem”, “Eixo das abscissas”, “Eixo das ordenadas”, “Primeiro quadrante”, “Segundo quadrante”, “Terceiro quadrante” e “Quarto quadrante. Veja exemplos de execução abaixo:

Entrada	Saída
3 6	Primeiro quadrante
3 -6	Quarto quadrante

104. [Reedição da 67] Você deve escrever um programa que produz os dois dígitos de verificação de um número de CPF. Para o cálculo do primeiro dígito verificador, os dígitos do número original, começando da direita para a esquerda (do menos significativo para o mais significativo) são multiplicados, respectivamente, por 2, por 3, por 4, e assim sucessivamente, até o primeiro dígito do número. O somatório dessas multiplicações é multiplicado por 10 e dividida por 11. O resto desta divisão (módulo 11) é o primeiro dígito verificador. Para calcular o próximo dígito, considera-se o dígito anterior como parte do número e efetua-se o mesmo processo. Por exemplo, considere o número 153245875. O primeiro digito seria calculado pela seguinte expressão: 10*(5*2 + 7*3 + 8*4 + 5*5 + 4*6 + 2*7 + 3*8 + 5*9 + 1*10) % 11, cujo resultado é 4. E o segundo digito, pela expressão 10*(4*2 + 5*3 + 7*4 + 8*5 + 5*6 + 4*7 + 2*8 + 3*9 + 5*10 + 1*11) % 11, cujo resultado é 0. Se uma das expressões dos digitos verificadores resultar em 10, o digito correspondente deve ser 0 (zero).

Entrada	Saída
153245875	40
352411573	01
635376221	63

105. Considere uma sequência de inteiros iniciada em um número natural qualquer e em que cada um dos termos seguintes é determinado em função do anterior, pela seguinte regra: 1) se o númeroanterior, Ni, for par, o número seguinte será Ni / 2; 2) se o numeroanterior, Ni, for ímpar, o número seguinte será 3 * Ni + 1. Um matemático chamado Collatz propôs que todas as sequências formadas através dessa regra, convergem para 1 (por isso, ficou conhecida por conjectura de Collatz). Pede-se que você escreva um programa que produza essa sequência de números a partir de um dado valor de N0, até produzir o número 1.

Entrada	Saída
3	3 10 5 16 8 4 2 1

106. Escreva um programa que, garantidamente, lê do usuário dois números inteiros entre 0 e 100 (ambos incluídos) e imprime sua soma. Se o usuário digitar um valor inválido, o programa deve pedir novamente o valor, indicando que o valor digitado é inválido. Veja exemplo de execução (baixe o tst correspondente -- http://db.tt/JV3dhwd7).

```
$ python somavalores.py
Número 1: 150
Número 1 inválido. Por favor, digite novamente.
Número 1: 95
Número 2: -5
Número 2 inválido. Por favor, digite novamente.
Número 2: 20
Soma: 115
```

Entrada	Saída
150 95 -5 20	Número 1: Número 1 inválido. Por favor, digite novamente. Número 1: Número 2: Número 2 inválido. Por favor, digite novamente. Número 2: Soma: 115

107. Os professores de programação 1 da UFCG mantêm uma tabela com o número de exercícios resolvidos por cada aluno e os nomes dos respectivos alunos. Assim, na posição de índice i da lista de exercícios temos o número de exercícios resolvidos pelo aluno da posição de índice i da lista de nomes. Pede-se que você escreva um programa que identifique o N-ésimo aluno da lista que resolveu no máximo K exercícios. O programa deve ler da primeira linha, a lista de exercícios resolvidos e da segunda linha os nomes dos alunos. Nas linhas seguintes, o programa deve ler uma série de Ns e Ks, para os quais deve imprimir na saída o N-ésimo aluno que resolveu no máximo K exercícios, bem como o número de exercícios resolvidos. Se não houve tal aluno, o programa deve imprimir apenas "-". O programa termina ao ler N e K iguais a zero. Espera-se que você faça o programa sem as funções de listas que são específicas da linguagem python, tais como max, min, sort, find, index, etc.

Entrada	Saída
7 6 8 5 4 8 5 9 4 2 8 a b c d e f g h i j k 1 5 2 5 5 5 6 5 1 6 0 0	d 5 e 4 j 2 - b 6

108. Escreva um programa que lê várias palavras da entrada e imprime a quantidade de letras maiúsculas em cada palavra lida. O programa para de ler palavras da entrada ao ler a palavra “fim”, que não deve ser processada. Veja exemplo de entrada e saída.

Entrada	Saída
ProBLemA casa SPLab TGV tvA fim	4 0 3 3 1

109. Escreva um programa que leia uma série de números inteiros não negativos e imprima o maior e o menor número lidos. Assuma que a entrada informará pelo menos dois números. O programa deve parar de ler inteiros ao ler o número -1. Não use: sort, max e min. Veja exemplos de entrada e saída.

Entrada	Saída
10 6 7 15 8 -1	maior: 15 menor: 6

110. Um estudante do curso de Computação da UFCG passou na seletiva do CNPQ para receber uma bolsa de R$500,00 mensais, com duração de um ano. Dadas as despesas mensais do estudante, e sabendo que seu saldo nunca ficou negativo, faça um programa que imprima as três primeiras letras do mês em que ele terminou com mais dinheiro.

Caso haja empate, considere o último mês. O programa recebe da entrada uma sequência de despesas mensais correspondente a doze meses. Considere que o estudante começou a receber a bolsa em janeiro. Veja exemplos de execução.

Entrada	Saída
200 300 300 900 800 100 200 500 200 300 1500 600	out
200 300 300 900 800 100 200 500 200 300 500 600	nov

111. Escreva um programa que lê 3 números e imprime na saída a quantidade de números iguais. Veja alguns exemplos de entrada e saída.

Entrada	Saída
3 -7 2	0
5 5 5	3
-1 4 4	2

112. Escreva um programa que informe o nome das pessoas que atingiram a maioridade penal (idade >= 18). O programa recebe, na primeira linha da entrada, uma lista de nomes de pessoas separados por espaço. Similarmente, na segunda linha, recebe a idade destas pessoas. O programa imprime na saída o nome das pessoas que atingiram a maioridade penal, na mesma ordem em que foram recebidos na entrada. Assuma que a quantidade de pessoas e de idades será sempre igual. Caso não haja pessoas 'maiores de idade' o programa não deve imprimir nada na saída. Veja os exemplos:

Entrada	Saída
Jansen Italo Ana 14 21 60	Italo Ana
Cristina Larissa 10 11	
Laura 18	Laura

113. Escreva um programa que utiliza a função conta_coincidencias(lista1, lista2) que recebe duas listas e que retorna a quantidade de elementos iguais que estão na mesma posição nas duas listas. Veja os asserts da função:

```
assert conta_coincidencias([1,2,3,4], [5,6,7]) == 0
assert conta_coincidencias([1,2,3,4], [2,3,3,4]) == 2
assert conta_coincidencias([1,2],[1,3]) == 1
```

O programa recebe, na primeira linha da entrada, a primeira lista de elementos. A segunda lista de elementos é recebida na segunda linha. Na saída, o programa imprime o valor retornado pela função conta_coincidências(lista1, lista2). Veja os exemplos de execução:

Entrada	Saída
1 2 3 4 5 6 7	0
1 2 1 3	1
1 2 3 4 2 3 3 4	2

114. Escreva um programa que utiliza a função encontra_menores(num, lista) que receba um número inteiro e uma lista de inteiros. A função deve retornar o primeiro valor da lista que seja menor que o número ou -1, se não houver valor que atenda à condição. Veja os asserts abaixo:

```
lista1 = [100,200,300,400]
```

```
lista2 = [1,3,5,7,9,11]
assert encontra_menores(100, lista1) == -1
assert encontra_menores(4, lista2) == 1
```

O programa recebe, na primeira linha da entrada, a lista de números. Na segunda linha o número inteiro é recebido. Na saída, o programa imprime o valor retornado pela função encontra_menores(num, lista). Veja os exemplos de execução:

115. Escreva um programa que utiliza função z_inicial(lista) que receba uma lista de palavras. A função deve retornar a quantidade de palavras da lista que iniciam com z. Assuma que as palavras não serão acentuadas nem terão sinais de pontuação. Veja os asserts:

```
lista1 = ["zumbi",”Zeca","Recife"]
lista2 = ["livro", "cd", "software"]
assert z_inicial(lista1) == 2
assert z_inicial(lista2) == 0
```

O programa recebe da entrada a lista de palavras. Na saída, o programa imprime o valor retornado pela função z_inicial(lista). Veja os exemplos de execução:

Entrada	Saída
zumbi Zeca Recife	2
livro cd Software	0

116. Escreva um programa que utiliza a função tem_vogais_adjacentes(palavra) que receba uma palavra e identifica se há vogais adjacentes na palavra. Se houver, retorne "sim" caso contrário, "nao". Assuma que as palavras não serão acentuadas nem terão sinais de pontuação. Veja os asserts:

```
assert tem_vogais_adjacentes("orfeu") == "sim"
assert tem_vogais_adjacentes("brasil") == "nao"
assert tem_vogais_adjacentes("voo") == "sim"
```

O programa recebe da entrada a palavra. Na saída, o programa imprime o valor retornado pela função tem_vogais_adjacentes(palavra). Veja os exemplos de execução:

Entrada	Saída
orfeu	sim
brasil	nao

117. Escreva um programa que leia na primeira linha da entrada um valor inteiro num. Na segunda linha da entrada, deverá ler uma lista de valores inteiros. Se num aparecer mais de uma vez na lista, o programa deve imprimir o índice da última ocorrência. Se nenhum valor atende ao critério, o programa imprime -1. Veja os exemplos abaixo.

Entrada	Saída
10 10 20 30 10	3
20 10 20 30 10	-1

118. Em um estudo meteorológico são lidos os valores de uma variável ao longo do tempo, em diversos momentos. Pede-se que você escreva um programa que identifique o menor e o maior valores observados, bem como as horas em que cada uma das observações ocorreu. O programa deve ler da entrada o valor da variável física na primeira linha da entrada. Cada valor é separado do outro, por um espaço. Da segunda linha da entrada, o programa deve ler as horas em que cada observação foi realizada. A saída deve indicar o menor e o maior valor lido, bem como as horas de cada uma das observações. Observe a entrada e a saída como exemplos de formatação a ser seguida. IMPORTANTE: você não deve usar max(), min() ou qualquer forma de ordenação para resolver esta questão.

Entrada	Saída
106.74 106.94 107.00 106.87 106.69 106.68 106.57 106.77 00:00 00:02 00:04 00:08 00:12 00:20 00:24 00:30	Min: 00:24 106.57 Max: 00:04 107.00

119. Escreva um programa que recebe uma data e retorna a data referente ao dia seguinte. Para simplificar, vamos assumir que não existe ano bissexto e, portanto, o mês de fevereiro tem 28 dias. Veja exemplos de entrada e saída.

Entrada	Saída
15 04 1974	16 04 1974
28 02 1980	01 03 1980

120. Os cartões de crédito com chip (smartcards) requerem de seus usuários uma senha numérica para que a transação financeira seja efetivada. Esta senha é fornecida para os usuários pela administradora de cartões e pode variar de tamanho dependendo da bandeira ou banco do cliente. As senhas obedecem uma lei de formação com o objetivo de torná-las mais seguras. Uma senha segura é composta por um número cujos algarismos nas posições ímpares são todos ímpares e que os algarismos nas posições pares são todos pares. Por exemplo: Considere a senha segura 7852. O primeiro algarismo (7) e o terceiro algarismo (5) são ímpares. O segundo algarismo (8) e o quarto algarismo (2) são pares. O seu programa deve receber uma senha numérica e verificar se essa senha é: segura ou insegura. Assuma que a senha tem, pelo menos, dois algarismos. Veja mais exemplos de execução:

Entrada	Saída
125638	segura
12346	insegura

121. Escreva um programa que calcula a associação em paralelo entre dois resistores R1 e R2 recebidos a partir da entrada padrão. Seu programa deve receber uma quantidade não determinada de pares de valores e só deve encerrar quando a resistência resultante for igual a zero. Considere apenas valores inteiros. A fórmula para o cálculo da resistência é R = (R1 * R2) / (R1 + R2). Veja o exemplo de execução.

Entrada	Saída
2 2 1 2	1 0

122. Escreva um programa que receba a partir da entrada padrão uma quantidade não determinada de números inteiros maiores que zero. Seu programa deve calcular a quantidade de números pares e ímpares, a média dos números pares e ímpares e a média geral dos números lidos (considere uma precisão de uma casa decimal para as médias). Seu programa deve parar quando o valor zero for lido. O zero é apenas uma sentinela e deve ser descartado. Veja o exemplo de execução.

Entrada	Saída
4 3 6 8 11 0	pares: 3 impares: 2 media pares: 6.0 media impares: 7.0 media geral: 6.4

123. Escreva um programa que leia da entrada uma sequência de números inteiros e que imprima a soma e a média dos valores lidos. O fim da sequência de valores é demarcada por um número negativo (que não é parte da sequência). Veja o exemplo de para entender como são formatadas a entrada e saída.

Entrada	Saída
10 8 4 -1	soma = 22 media = 7.33

124. Um sistema automatizado de tiro ao alvo captura o local em que um jogador atira na forma de um ponto em um plano cartesiano. A mosca do alvo é representada pela origem do plano -- posição (0.0, 0.0). Pede-se que você faça o programa que controla o jogo. O jogador pode disparar quantas vezes desejar. A cada disparo, o sistema recebe a posição em que o jogador atirou (um par de valores separados por vírgulas que representam a posição no plano) e

imprime a distância a que o tiro chegou do alvo. Para encerrar o jogo, o jogador deve disparar a mais de 200 centimetros de distância da mosca. Ao final, o sistema deve indicar o número total de disparos válidos e a distância média dos tiros dados. Veja o exemplo para saber como são formatadas a entrada e saída.

Entrada	Saída
85, 45 35.0, 40.0 -20, 15 215, 181	96.18 53.15 25.00 -- num disparos: 3 distancia media: 58.11

125. Escreva um programa que contabiliza o número de palavras iniciadas com vogais. O programa deve ler uma sequência de palavras da entrada até que a palavra "fim" seja lida. Ao final, deve indicar o número total de palavras, quantas das palavras são iniciadas por vogais e que percentagem isso representa. Veja o exemplo para entender como devem ser formatadas a entrada e a saída.

Entrada	Saída
quantas destas palavras sao iniciadas por vogais fim	total de palavras: 7 iniciadas por vogais: 1 (14.29%)

126. Um jogo simples (e tolo) que se costuma fazer quando se começa a programar computadores é o "adivinhe o número". A idéia é que o computador sorteia um número natural qualquer até 1000 e o usuário tenta adivinhá-lo através de sucessivas tentativas. A cada tentativa, o programa indica se o número a ser adivinhado é menor, maior ou se o jogador o acertou. Ao acertar o número, o programa diz quantas jogadas foram realizadas. Pede-se que você implemente o jogo. Veja o exemplo de execução abaixo, para ter idéia do funcionamento.

Entrada	Saída
400 120 50 80 70 60 54 56	Adivinhe o número... o número é menor... o número é menor... o número é maior... o número é menor... o número é menor... o número é menor... o número é maior... Acertou em 8 tentativas.

127. Jack é um dos melhores funcionários da transportadora onde ele trabalha. Ele sempre está inovando através da busca de soluções para a redução de custos na empresa. Atualmente, Jack tem observado que um dos pontos fracos da empresa é o setor de carga e descarga de caminhões porque só há uma única plataforma para este propósito. Jack está com a ideia de desenvolver um programa que determina quanto tempo a plataforma fica ociosa durante o dia e você deve ajudá-lo nesta tarefa.

Entrada

A entrada inicia-se com um inteiro N > 1, indicando o número de carregamentos efetuados no dia. Em seguida, as próximas N linhas são compostas por dois inteiros não negativos que indicam, respectivamente, os horários de início e de conclusão de cada carregamento.

Saída

Seu programa deve imprimir na saída o tempo total em que a plataforma de carregamento ficou ociosa no dia.

Observações

O expediente na empresa sempre começa a partir do primeiro carregamento e termina quando o último carregamento é concluído. Além disso, os horários de início e término de um carregamento são sempre exatos, um carregamento sempre inicia e termina num mesmo dia e os horários de cada carregamento são dados em ordem.

Entrada	Saída
3 2 7 14 19 19 23	7

128. Um meteorologista precisa classificar valores de temperatura em certo período de tempo, em relação à média histórica. Escreva um programa que leia a média histórica de temperatura, e em seguida uma sequência de medições de temperaturas relativas a certo período de tempo e que imprima quantas medições foram iguais ou acima da média e quantas foram abaixo (bem como as respectivas médias de cada grupo). Uma medição com valor maior ou igual a 100.0 indica o fim da série (esse valor é um sentinela e deve ser desconsiderado). A média histórica é o primeiro valor lido da entrada. Os demais valores são as medições. Veja o exemplo, para entender como são formatadas a entrada e a saída do programa.

Entrada	Saída
20.0 22.6 23.2 21.0 19.5 100	média histórica considerada: 20.0 medições acima da média: 3 (75.0%) medições abaixo da média: 1 (25.0%)

129. Um meteorologista precisa de um programa que detecta mudanças bruscas de temperaturas. Pede-se que você escreva um programa que, dada uma sequência de temperaturas, pare ao encontrar o primeiro par em sequência cuja diferença seja acima de 2.0 graus celsius. Veja o exemplo de entrada e saída.

Entrada	Saída
20.0 20.5 20.6 20.8 21.9 24.0	dif: 0.50 (ok) dif: 0.10 (ok) dif: 0.20 (ok) dif: 1.10 (ok) dif: 2.10 (diferenca acima de 2.0 detectada)

130. Escreva um programa que realiza as operações básicas de uma calculadora: adição, subtração, multiplicação e divisão. Seu programa deve realizar uma quantidade não determinada de operações. Cada linha da entrada possui o código de operação (1 – adição, 2 – subtração, 3 – multiplicação, 4 – divisão, 5 – Sair) e os dois operandos, todos inteiros. Considere apenas valores inteiros na impressão dos resultados. Quando a operação 5 for informada, o seu programa deve encerrar. Veja o exemplo de execução.

Entrada	Saída
1 2 5 4 4 3 5	7 1

131. Escreva um programa que leia o nome do cliente e o saldo inicial de sua conta bancária. Em seguida, seu programa deve ler uma quantidade não determinada de pares de valores que indicam respectivamente o código da operação (1 – Sacar, 2 – Depositar, 3 – Encerrar) e o valor envolvido. Quando o código 3 for informado, o seu programa deve encerrar e imprimir o saldo final da conta. Considere uma precisão de duas casas decimais. Veja o exemplo de execução.

Entrada	Saída
Wilkerson 100.00 1 50.00 2 20.00 3	Saldo de R$ 70.00 na conta de Wilkerson

Matheus 20.00 1 100.00 3	Saldo de R$ -80.00 na conta de Matheus

132. No jogo de guerra de cartas, ganha uma rodada aquele jogador que tiver a carta de maior valor e ganha o jogo aquele que tiver vencido o maior número de rodadas. Você deve construir um programa que simule este jogo. Cada rodada será representada por uma linha com dois inteiros positivos separados por espaço. O primeiro inteiro representa o valor da carta utilizada pelo primeiro jogador. Já o segundo inteiro representa a carta do segundo jogador. Seu programa deve encerrar a partida quando receber as duas cartas com valor zero. Terminada a partida, seu programa deve imprimir uma linha com a mensagem: "Jogador 1: X, Jogador 2: Y, Empates: Z". X e Y representam o número de vitórias do primeiro jogador e do segundo jogador respectivamente. Já Z é o número de rodadas empatadas (cartas do mesmo valor). Veja o exemplo.

Entrada	Saída
12 1 11 11 2 3 3 4 4 5 13 1 0 0	Jogador 1: 2, Jogador 2: 3, Empates: 1

133. Um robô é colocado na posição (0, 0) do plano cartesiano. Este robô apresenta 4 possibilidades de movimento: C (cima), B (baixo), E (esquerda), D (direita). Caso ele se mova para cima, sua nova posição será (0, 1). Caso ele se mova para baixo, ele irá para a posição (0, -1). Se for para esquerda, assume a posição (-1, 0) e, caso vá para direita, a posição (1, 0). Construa um programa que irá receber, em cada linha, um comando de movimento (C, B, E ou D), seguido de um número inteiro não-negativo de unidades de movimento. Considere que o robô começa na posição (0, 0). Como exemplo, o comando "C 5" colocaria o robô na posição (0, 5). Seu programa deve executar até que o número de unidades de movimento seja zero. É apenas neste momento que o programa imprime uma linha com as coordenadas (separadas por um espaço) do robô no plano. Veja o exemplo abaixo:

Entrada	Saída
B 1 B 2 D 1 D 4 E 0	5 -3

134. Numa empresa, cada funcionário é identificado por um código composto de uma letra em maiúscula seguida por 5 dígitos. Um exemplo seria o código A12345. Funcionários que desempenham a mesma função são identificados pelo caractere inicial do código. Pede-se que você faça um programa que controle o ponto dos funcionários desta empresa. Seu programa recebe o registro de ponto do funcionário em uma linha que começa com o caractere R seguida do código do funcionário. Exemplo: "R A12345". Seu programa também deve permitir contabilizar o número de registros de pontos realizados por funcionários de determinada categoria. Esta ação é marcada por uma linha que começa com o caractere P seguido da letra em maiúsculo que representa esta categoria. Exemplo: "P A". Observe que um mesmo funcionário pode registrar mais de uma vez o ponto. E cada registro deste mesmo funcionário deve ser contabilizado no total de cada categoria. Seu programa deve encerrar ao encontrar uma linha com o caractere "S". Veja o exemplo de uso.

Entrada	Saída
R A12345 R A12345 P A R A00007 P A R B90000 P B R B90001 P B S	2 3 1 2

135. Escreva a função prevogais(palavra) que retorne a lista de todos os caracteres que precedem vogais de uma palavra lida da entrada.

```
assert prevogais("exemplo") == ['x', 'l']
assert prevogais("hiato") == ['h', 'i', 't']
```

136. Escreva uma função que retorna a lista dos divisores próprios de um dado número. Os divisores próprios de um número n são os números m que dividem n excluindo ele mesmo. Veja o assert abaixo:

```
assert divisores_proprios(12) == [1, 2, 3, 4, 6]
```

137. Escreva um programa que leia da entrada uma série de temperaturas (em Celsius ou Fahrenheit) e que imprima a temperatura convertida correspondente (em Fahrenheit ou Celsius). O fim da série de temperaturas é demarcada pela string "fim". Seu programa deve necessariamente conter duas funções que devem ser usadas para fazer as conversões. A função celsiusToFahrenheit(t) recebe um valor de temperatura em Celsius e retorna a temperatura correspondente em Fahrenheit. A função fahrenheitToCelsius(t) recebe um valor de temperatura em Fahrenheit e retorna a temperatura correspondente em Celsius. Lembrando que Tc/5 = (Tf - 32)/9. Veja o exemplo para entender como são formatadas a entrada e saída.

Entrada	Saída
F 35.7 F 32 C 15.2 fim	C 2.1 C 0.0 F 59.4

138. O peso ideal de uma pessoa é calculado com base no sexo e na altura desta pessoa. Para homens, o peso ideal é calculado através da fórmula "72.7 * altura - 58". Para mulheres, a fórmula é "62.1 * altura - 44.7". Escreva um programa que lê da entrada uma série de dados (sexo e altura) de pessoas e imprime o valor do peso ideal. O programa deve parar de calcular o peso ideal quando a string "****" for lida. O programa deve conter NECESSARIAMENTE a função calculaPeso(sexo, altura) que recebe o sexo e a altura da pessoa e retorna o peso ideal que foi calculado. Veja o exemplo para entender como são formatadas a entrada e saída.

Entrada	Saída
m 1.89 ****	Homem: peso ideal é 79.4
F 1.60 ****	Mulher: peso ideal é 54.7
M 1.60 ****	Homem: peso ideal é 58.3

139. Escreva um programa que lê da entrada uma série de dados de figuras geométricas (quadrado, círculo ou triangulo) e calcula as respectivas áreas. Se a figura é um quadrado o valor do lado é informado. No caso de um círculo temos o valor do raio e, por fim, se a figura for um triangulo, temos as informações de base e altura. O programa deve parar de calcular as áreas quando a string "fim" for lida. O programa deve conter NECESSARIAMENTE três funções: areaQuadrado(lado), areaTriangulo(base, altura) e areaCirculo(raio) que retornam os valores das áreas das figuras correspondentes. Veja o exemplo para entender como são formatadas a entrada e saída.

Entrada	Saída
Q 2 fim	A área do quadrado é 4.00
C 2.5 fim	A área do círculo é 19.63
T 2.3 5 fim	A área do triangulo é 5.75

140. Escreva a função letras_alternadas que retorna uma string formada com as letras em posições alternadas, começando pela primeira, de uma dada palavra.

```
assert letras_alternadas("casa") == "cs"
assert letras_alternadas("exemplo") == "eepo"
```

141. Escreva a função conta_letra(letra, frase) que receba como parâmetro uma letra e uma frase, ambas string. A função deve retornar a quantidade de vezes que a letra informada aparece na frase. Considere que as palavras não serão acentuadas. Veja o assert abaixo:

```
assert conta_letra("a", "Abacaxi e laranja") == 6
```

142. Escreva um programa que receba um código gerado para acesso a um formulário online e verifica se ele é "verdadeiro" ou "falso". Um código verificado como "verdadeiro" é composto por uma sequência de dígitos que se alternam entre pares e ímpares ou entre ímpares e pares. Não são admitidos pares ou ímpares juntos. Assuma que pelo menos 2 dígitos compõem o código. O código 7852 é verificado como verdadeiro: (7) é ímpar; (8) é par; (5) é ímpar e (2) é par. A saída do programa deve informar se o código é 'verdadeiro' ou 'falso' e o número de algarismos do código.

Entrada	Saída
25638	verdadeiro: 5 algarismos.
77	falso: 2 algarismos.

143. Escreva a função lanchemaispedido(pedidos) que retorna o elemento mais frequente da lista pedidos. Considere que o elemento mais frequente é aquele que aparece mais da metade das vezes na lista. Sua função deve funcionar conforme os asserts abaixo.

```
ines = ['tapioca','tapioca','salada','bolo','misto','tapioca', 'tapioca']
marcos = ['suco','coxinha','suco','misto','folhado']

assert lanchemaispedido(ines) == 'tapioca'
assert lanchemaispedido(marcos) == None
```

144. No jogo "zerinho ou um americano", cada um dos participantes indica ao mesmo tempo com as mãos uma escolha entre 1 e 10. Em seguida, todos os valores escolhidos pelos participantes são somados e inicia-se uma contagem circular a partir da primeira pessoa. A contagem é realizada até chegar ao valor da soma definido inicialmente. O vencedor é exatamente a pessoa em que a contagem foi encerrada.

Entrada

A entrada é composta por valores de várias partidas, cada uma em uma linha. Cada partida é composta pelos valores indicados por cada participante da partida, onde cada valor corresponde a um número inteiro entre 1 e 10. Assuma que em uma partida de “zerinho ou um americano” haverá no mínimo 2 jogadores. O programa é encerrado quando um 0 (zero) for encontrado. O último valor lido é utilizado apenas como critério de parada e deve ser descartado.

Saída

Seu programa deve imprimir na saída a posição do participante vencedor em cada uma das partidas.

Entrada	Saída
3 5 4 1 2 4 1 0	5 1

145. [TOPPL1-20122]Escreva um programa que recebe uma sequência de palavras até ler "***". O programa deve imprimir o total de palavras começadas por consoante e que tenham pelo menos 5 letras.

Entrada	Saída
asa controle janela ***	2
boi controle Janela alvorada Escudo TCPIP	3

146. [TOPPL1-20122] Escreva a função **soma_vizinhos** que recebe uma matriz, representada na forma de lista de listas e dois parâmetros **lin** e **col** representando a linha e a coluna de um elemento da matriz (a numeração da matriz começa na linha e coluna 1). Tal função deve retornar a soma do elemento da posição lin, col com os elementos adjacentes (superior, inferior, à esquerda e à direita) que existirem. Veja o exemplo:

```
matriz = [
    [ 1,  2,  3],
    [ 8, 10, 12],
    [21, 24, 27],
]
assert soma_vizinhos(matriz, 2, 2) == 56
assert soma_vizinhos(matriz, 1, 1) == 11
```

147. [TOPPL1-20122]Apenas estudantes que tenham se inscrito e que tenham média acima de certo valor mínimo podem fazer o TOPPL. Pede-se que você escreva a função **filtra_alunos()** que seleciona os alunos aptos a fazer a prova de acordo com essas condições. O primeiro parâmetro recebido pela função é uma lista dos alunos da disciplina com suas respectivas médias, na forma de lista de pares de matrículas e médias. O segundo é a lista dos inscritos, na forma de lista de matrículas. O terceiro parâmetro é a média mínima necessária para fazer o TOPPL. A função deve ter efeito colateral. Ela deve alterar o conteúdo da lista de alunos, eliminando dela todos os alunos que não estão aptos a fazer a prova, seja por não ter média suficiente, seja por não ter feito a inscrição. Finalmente, a função deve retornar o número de alunos que foram eliminados da lista. Veja os asserts para compreender melhor a semântica da função.

```
inscritos = [121, 123, 124]
alunos = [ (120,8.0), (121,7.5), (122,5.0), (123,6.0), (124,9.0), (125,4.0) ]
assert filtra_alunos(alunos, inscritos, 7.0) == 4
assert alunos == [ (121,7.5), (124,9.0) ]
```

148. [TOPPL1-20122]No início de cada semestre letivo, o aluno deve fazer sua matrícula e pode escolher as disciplinas que deseja cursar. Ao escolher uma disciplina, o aluno já deve ter cursado com sucesso as disciplinas que são pré-requisitos da disciplina escolhida. A grade de disciplinas de um curso pode ser representada por um dicionário em que cada chave é o nome da disciplina e o seu valor associado é a lista de pré-requisitos efetivos da disciplina. Crie a função disciplinas(grade, disciplinas_cursadas) que recebe como primeiro parâmetro um dicionário com a grade das disciplinas do curso e todos os pré-requisitos associados a cada uma delas, como segundo parâmetro recebe uma lista das disciplinas já cursadas por um aluno. A função deve retornar uma lista com as disciplinas nas quais o aluno pode se matricular. Veja os asserts abaixo:

```
grade = {"p1": [],
       "lp1": [],
       "ic": [],
       "calc1": [],
       "p2": ["ic", "p1", "lp1"],
       "lp2": ["ic", "p1", "lp1"],
       "grafos": ["ic", "p1", "lp1"],
       "calc2": ["calc1"],
       "edados": ["ic", "p1", "lp1", "p2", "lp2", "grafos"],
       "leda": ["ic", "p1", "lp1", "p2", "lp2", "grafos"]}

assert set(disciplinas(grade, [])) == set(["ic", "p1", "lp1", "calc1"])
assert set(disciplinas(grade, ["ic", "p1", "lp1", "calc1"])) == set(["p2",
"grafos", "calc2", "lp2"])
```

149. [TOPPL1-20122]Uma sequência de DNA é uma série de pelo menos quatro letras representando a estrutura primária de uma molécula ou cadeia de DNA. As letras possíveis são A, C, G e T, representando os quatro nucleotídeos existentes. Um dos objetivos de um projeto de mapeamento genético é encontrar ocorrências de uma pequena cadeia de DNA dentro de uma outra cadeia maior. Escreva um programa que indica as posições em que uma cadeia menor ocorre dentro de uma cadeia maior.

Entrada

A entrada é composta de duas linhas: (1) a primeira linha de entrada contém a primeira cadeia de DNA de tamanho N cujos nucleotídeos são separados por espaços; (2) a segunda linha da entrada contém a segunda cadeia de DNA de tamanho M (onde M >= N).

Saída

Seu programa deve imprimir em linhas separadas as posições em que a primeira cadeia de DNA ocorre dentro da segunda cadeia.

Veja um exemplo de execução abaixo e observe que a posição inicial da cadeia é 1:

Entrada	Saída
C A G T A T C A G T T C C A G T	3 9

150. Escreve um programa que calcule o fatorial de um número n fornecido (0 <= n <= 200), e imprima a quantidade de zeros no final desse número. O programa deve conter duas funções distintas, uma para calcular o fatorial e outra para contar a quantidade de zeros. Não devem ser usadas funções do python para calcular o fatorial nem a quantidade de zeros.

Entrada	Saída
5	1
20	4

151. As senhas criadas por pessoas geralmente remetem a uma palavra ou frase familiar que é modificada de alguma forma a fim de torná-la difícil de decifrar. Os cadastros em formulários na Web costumam testar a segurança da senha caracterizando-a, por exemplo, como "forte" ou "fraca". Pede-se que você crie um programa que receba palavras e o nível de segurança ("forte" ou "fraco") desejado e gere e imprima senhas. O programa deve parar quando ler o marcador "***". Sua implementação deve conter a função criaSenhaFraca(palavra) que recebe a palavra informada pelo usuário e a utiliza para gerar a senha. A senha é formada pela palavra compreendida entre dois caracteres abre parêntesis e dois caracteres fecha parênteses. O programa deve conter, ainda, a função criaSenhaForte(palavra) que recebe a palavra informada pelo usuário, cria uma senha fraca (de acordo com a explicação anterior) e depois substitui caracteres de acordo com a seguinte política:

```
o,O -> 0
i,I,L,l -> 1
e,E -> 3
a,A -> 4
b,B -> 6
t,T -> 7
```

Veja exemplos de entrada e saída para entender como o programa interage com o usuário.

Entrada	Saída
alo fraco alo forte ***	((alo)) ((410))
antonio forte ***	((4n70n10))

152. Na computação, os dados são armazenados utilizando a representação binária. Os números em binário são compostos por uma sequencia de zeros e uns conhecidos como bits. O tamanho do número, ou seja, a quantidade de bits, depende do espaço de memória reservado para o seu armazenamento. Por exemplo, armazenar o número 5 em um espaço de 8 bits, significa: 1)Encontrar a representação binária de 5 que é 101. 2)Preencher com zeros a esquerda a quantidade de bits necessárias para chegar a 8. O valor esperado é 00000101. Pede-se que você crie um programa que receba números decimais, converta-os para binário de tamanho padrão 8 bits e informe quantos zeros ou uns existe no número. Seu programa deve receber da entrada, linhas contento o decimal (0 menor que d menor que 128) a ser convertido seguido de 0 ou 1(que irá indicar se o programa deve contar o número de zeros ou uns). Na saída, deve imprimir a representação binária do número no tamanho de 8 bits seguido do número de zeros ou uns (de acordo com o requerido na entrada) contidos nesta representação. O programa para de executar quando o usuário informar na entrada a sequencia ***. Você pode usar a função bin(), nativa de Python, para calcular a representação binária do número decimal. Veja exemplos de entrada e saída para entender como o programa funciona.

Entrada	Saída
7 1	00000111 3

7 0 ***	00000111 5
5 1 5 0 ***	00000101 2 00000101 6

153. O tabuleiro do jogo LÍNEA é composto por n casas adjacentes. Cada jogador começa da casa zero e em cada casa há uma indicação para qual casa o jogador deve ir. Vence o jogador que terminar na casa 0. Neste exercício, o tabuleiro é representado por uma lista com n inteiros. Cada posição da lista representa uma casa do tabuleiro e o elemento dessa lista representa a próxima casa para a qual o jogador deve ir. Faça uma função eh_vencedor_linea(tabuleiro) que retorna se o jogador irá vencer ou não uma partida. Caso o jogador saia do tabuleiro (precise ir para uma casa que não existe), ele imediatamente perde o jogo. Caso o jogador retorne para a posição de origem (primeira casa), ele é decretado como vencedor. Considere que nenhum tabuleiro fará o jogador passar pela mesma casa mais de uma vez. Veja os asserts:

```
assert eh_vencedor_linea([2,0,1])
assert not eh_vencedor_linea([1,2,3])
assert eh_vencedor_linea([1,2,3,4,5,0])
```

154. Para os computadores, tudo é número binário. Podemos representar números inteiros, em binário, utilizando notações como Excesso de n e Complemento de 1.
Na representação binária, o tamanho do número, ou seja, a quantidade de bits, depende do espaço de memória reservado para o seu armazenamento. Por exemplo, armazenar o número 5 em um espaço de 8 bits, significa: 1)Encontrar a representação binária de 5 que é 101. 2)Preencher com zeros a esquerda a quantidade de bits necessárias para chegar a 8. O valor esperado é 00000101.
Para representar um número decimal na notação excesso de n, deve-se adicionar n a este decimal e converter este resultado para binário no tamanho requerido.
A representação de um número decimal positivo em complemento de 1 é identica à sua representação em binário. Caso o número seja um decimal negativo, deve-se primeiro obter a sua representação em binário no tamanho requerido e posteriormente inverter todos os bits (onde há 0, substitui por 1 e vice-versa).
Pede-se que você crie um programa que receba números decimais e represente-os na notação Excesso de 127 ou Complemento de 1 em tamanho de 8 bits. Seu programa deve receber na entrada, linhas com "E127" (que significa Excesso de 127) ou "C1" (que significa Complemento de 1) seguido de um decimal (maior que -128 e menor que 128) a ser convertido para binário e representado no esquema informado. Na saída, deve imprimir a representação binária do número, no tamanho de 8 bits, no esquema requerido. O programa para de executar quando o usuário informar na entrada a sequencia ***. O seu programa deve NECESSARIAMENTE conter duas funções excesso_127(numero) e complemento1(numero). Você pode usar a função bin(), nativa de Python, para calcular a representação binária do número decimal. Veja exemplos de entrada e saída para entender como o programa funciona.

Entrada	Saída
C1 7 E127 7 ***	00000111 10000110
C1 5 ***	00000101

155. Escreva a implementação de uma função sequencia_caras(lancamentos), que recebe um vetor contendo a sequência de resultados de lançamentos de uma moeda e retorna o tamanho da maior sequência de '1's consecutivos. Veja exemplos de asserts:

```
jogo1 = [0,1,1,0,1,0,0,0]
jogo2 = [1,0,1]
jogo3 = [0,1,1,1,0]

assert sequencia_caras( jogo1 ) == 2
assert sequencia_caras( jogo2 ) == 1
assert sequencia_caras( jogo3 ) == 3
```

156. Escreva uma função que receba uma lista de números naturais e retorna o número que é dominante na lista. Um número é dominante se ele aparece em mais da metade da sequência. Caso não haja um dominante na sequência, a função deve retornar -1. Veja os asserts abaixo.

```
assert dominante([0,0,0,1,2,0,3]) == 0
```

```
assert dominante([10,10,10]) == 10
assert dominante([1,2,2,3]) == -1
```

157. Escreva a função filtra_altera_lista(num, lista) que receba um número inteiro e uma lista não vazia de inteiros não-negativos. A função deve alterar a lista removendo os elementos das posições que não são divisíveis por num. Considere que 0 é divisível por qualquer número e de que num > 0. Veja os asserts abaixo:

```
lista1 = [0,1,2,3,4,5,6]
lista2 = [2,3,5,7,11,13,17]

filtra_altera_lista(2, lista1)
assert lista1 == [0,2,4,6]
filtra_altera_lista(3, lista1)
assert lista1 == [0,6]

filtra_altera_lista(3, lista2)
filtra_altera_lista(2, lista2)
assert lista2 == [2, 17]
```

158. Escreva a função filtra_lista(num, lista) que receba um número inteiro e uma lista não vazia de inteiros não-negativos. A função deve retornar uma nova lista apenas com os elementos onde a posição do elemento na lista original é divisível por num. Considere que 0 é divisível por qualquer número e de que num > 0. Veja os asserts abaixo:

```
lista1 = [0,1,2,3,4,5,6]
lista2 = [2,3,5,7,11,13,17]
assert filtra_lista(2, lista1) == [0,2,4,6]
assert filtra_lista(3, lista1) == [0,3,6]
assert filtra_lista(4, lista2) == [2,11]
assert filtra_lista(40, lista2) == [2]
```

159. Escreva a função acima_de(N, L) que recebe um valor numérico N e uma lista de números L e que retorna uma lista de índices de L, correspondendo aos elementos de L maiores que N. Envie como resposta apenas função.

160. Escreva um programa que identifica os alunos que resolveram um número de exercícios acima da média da turma, usando a função acima_de(...) que você produziu anteriormente. O programa deve ler da primeira linha da entrada uma sequência de números de exercícios resolvidos pelos alunos. Da segunda linha, os nomes dos alunos. Na saída imprime apenas os que estão acima da média da turma. Você não deve usar sum(), max(), min() ou sort(). Você DEVE usar a função acima_de(N, L).

Entrada	Saída
10 12 20 16 15 19 22 a b c d e f g	c 20 f 19 g 22

161. Escreva a função conta_palavras(k, palavras) que receba como parâmetro um valor inteiro k e uma string com uma série de palavras separadas por ":". A função deve retornar a quantidade de palavras com comprimento maior ou igual a k. Considere a existência de pelo menos uma palavra na série de palavras informada e k >= 0. As palavras não são acentuadas nem têm caracteres especiais. (Dica: use a função split)

```
assert conta_palavras(5, "zero:um:dois:tres:quatro:cinco") == 2
```

162. Escreva a função maior_palavra(lista) que receba uma lista de palavras e retorne a palavra de maior comprimento. Se houver mais de uma palavra, qualquer uma pode ser retornada. Assuma que há, pelo menos, uma palavra na lista. As palavras não são acentuadas nem têm caracteres especiais. Veja os asserts abaixo:

```
lista1 = ["palavra", "exemplo", "computador", "mouse"]
lista2 = ["outra", "palavra", "como", "exemplo", "mouse"]
assert maior_palavra(lista1) == "computador"
assert maior_palavra(lista2) in ["palavra", "exemplo"]
```

163. Escreva a função divisor(num, lista) que receba um inteiro e uma lista de inteiros e retorne o índice do primeiro elemento da lista divisível pelo número ou -1 se não existir. Veja os asserts:

```
lista1 = [100,10,40,50]
lista2 = [3,15,50,23,5]
assert divisor(10, lista1) == 0
assert divisor(5, lista2) == 1
```

164. Escreva a função encontra_menores(num, lista) que receba um número inteiro e uma lista de inteiros. A função deve retornar o primeiro valor da lista que seja menor que o número ou -1, se não houver valor que atenda à condição. Veja os asserts abaixo:

```
lista1 = [100,200,300,400]
```

```
lista2 = [1,3,5,7,9,11]
assert encontra_menores(100, lista1) == -1
assert encontra_menores(4, lista2) == 1
```

165. A famosa feijoada da cantina de Dona Inês acontece toda quinta-feira no DSC. Os alunos fazem seus pedidos em grupos e em fila. Ou seja, chega um grupo de alunos que pede 10 feijoadas, depois aparece outro grupo que pede 5 feijoadas. No entanto, Dona Inês sempre prepara apenas n_feijoadas e a demanda por feijoadas é sempre maior ou igual ao que ela preparou. Quando a feijoada não é capaz de servir todos os alunos de um grupo, todos os alunos deste grupo desistem de comer feijoada. Mesmo com feijoadas sobrando, Dona Inês também para de servir almoço depois que o primeiro grupo de alunos desistir de comer em sua cantina. Faça uma função quantos_comeram(n_feijoadas, fila) que, dada uma fila de pedidos (sempre em grupo), retorne quantas feijoadas foram consumidas de fato. Veja os asserts abaixo para entender melhor a questão:

```
assert quantos_comeram(10, [10, 10]) == 10
assert quantos_comeram(12, [10, 10]) == 10
assert quantos_comeram(2, [10, 10]) == 0
assert quantos_comeram(5, [2, 3, 5]) == 5
```

166. Os alunos de Laboratório de Programação I do semestre 2012.2 foram divididos em 5 turmas. A alocação dos alunos nas turmas de laboratório é definida através de duas listas: uma contém o número da turma em que o aluno está matriculado e a outra contém os nomes dos alunos. Assim, o aluno da posição i numa lista está matriculado na turma da posição i na outra lista. Escreva a função cria_lista_presenca(turmas, nomes, turma) que receba as duas listas com números de turmas e respectivos nomes e o número da turma para a qual será gerada a lista de presença. Sua função deve retornar uma lista com os nomes dos alunos matriculados na turma indicada. Veja os asserts abaixo:

```
turmas = [1, 2, 2, 4, 5, 3, 5]
nomes = ["Maria", "Pedro", "Carlos", "Ana", "Carla", "Joao", "Jose"]
assert cria_lista_presenca(turmas, nomes, 5) == ["Carla", "Jose"]
```

167. Uma empresa aérea pretende mudar o serviço de embarque de seus passageiros em seus voos e decidiu que os passageiros serão chamados a embarcar de acordo com uma "fila virtual" na qual os passageiros são colocados, de acordo com a ordem de aquisição das passagens. Contudo, a empresa quer priorizar o embarque de pessoas idosas. Embora os idosos sejam priorizados, a empresa quer manter os idosos ma mesma ordem de entrada na fila. Pede-se que você implemente uma função que, dada uma lista das idades dos passageiros (ordenada segundo a fila de embarque), altere a lista da seguinte forma: o primeiro idoso encontrado na lista deve trocar de posição com o primeiro da fila, o segundo idoso deve trocar de posição com o segundo da fila, e assim por diante. Considere que idosos são pessoas com 60 anos ou mais. Veja o assert abaixo para compreender a especificação:

```
fila = [25, 33, 67, 61, 35, 8, 12, 15, 22, 63, 75, 30, 34]
idosos_inicio(fila)
assert fila == [67, 61, 63, 75, 35, 8, 12, 15, 22, 25, 33, 30, 34]
```

168. Um parque de diversões pretende alterar a política de acesso aos brinquedos para dar prioridade para menores de idade. Inicialmente a fila é organizada de acordo com a ordem de chegada. Pede-se que você implemente uma função que, dada uma lista das idades das pessoas (ordenada segundo a ordem de chegada), altere a lista da seguinte forma: percorrendo a lista de trás para a frente, o primeiro maior de idade encontrado na lista deve trocar de posição com o último da fila, o segundo maior de idade deve trocar de posição com o penúltimo da fila, e assim por diante. Considere que maiores de idades são pessoas com 18 anos ou mais. Veja o assert abaixo para compreender a especificação:

```
fila = [12, 21, 35, 8, 12, 15]
maiores_final(fila)
assert fila == [12, 12, 15, 8, 21, 35]
```

169. O número oposto de qualquer número n é um número que, se somado a n, resulta em 0. Por exemplo, -7 é o número oposto de 7 porque 7 + (-7) = 0. Escreva a função lista_so_com_oposto(lista) que recebe uma lista de inteiros e altera a lista recebida de forma a manter na lista apenas os elementos cujo oposto também está na lista. Veja os asserts abaixo.

```
lista1 = [1, 2, 1, 3, 4, -1, -3, 5]
lista2 = [-2, 2, 5, 3, -2]

lista_so_com_oposto(lista1)
assert lista1 == [1, 1, 3, -1, -3]

lista_so_com_oposto(lista2)
assert lista2 == [-2, 2, -2]
```

170. Escreva a função remove_palavras_com_menos_vogais(lista) que recebe uma lista de palavras e altera a lista recebida de forma a manter na lista apenas as palavras que possuem mais vogais do que consoantes.

```
lista1 = ['arara', 'bic', 'bacia']
```

```
remove_palavras_com_menos_vogais(lista1)
assert lista1 == ['arara',  'bacia']
```

171. Escreva a função remove_listas_com_todos_menores(lista, limite) que recebe uma lista de listas de inteiros e um valor limite e altera a lista recebida de forma a eliminar as listas em que todos os seus valores inteiros são menores do que o valor limite.

```
lista1 = [[10, 20], [1, 2, 3, 4], [-5, 20, 20]]



remove_listas_com_todos_menores(lista1, 11)
assert lista1 == [[10, 20], [-5, 20, 20]]
```

172. Escreva a função soma_mats(M1, M2) que retorna a matriz soma M1 + M2. Veja os asserts abaixo.

```
m1 = [[1,  3, 7],
      [4, -2, 0]]

m2 = [[10, 20, -7],
      [ 5, 14,  3]]

assert soma_mats(m1, m2) == [[11, 23, 0], [9, 12, 3]]
```

173. Escreva a função diag_principal(M) que retorna uma lista com os valores da diagonal principal de M, sendo M uma matriz quadrada.

174. Escreva a função diag_secundaria(M) que retorna uma lista com os valores da diagonal secundaria de M, sendo M uma matriz quadrada.

175. Escreva a função transposta(M) que recebe uma matriz M e retorna uma nova matriz igual à matriz transposta de M. M = [[1,1,1,1], [2,2,2,2], [3,3,3,3]] assert transposta(M) == [[1,2,3], [1,2,3], [1,2,3], [1,2,3]] assert M == [[1,1,1,1], [2,2,2,2], [3,3,3,3]]

176. Escreva a função coluna(matriz, i) que retorna a coluna de índice i da matriz, na forma de uma lista de valores.

177. Escreva a função somacols(M) que retorna uma lista das somas dos valores da matriz por coluna.

178. Escreva a função eh_triangular_sup(mq) que retorna True se a matriz quadrada mq é triangular superior (todos os elementos abaixo da diagonal principal são iguais a zero). A matriz mq é passada na forma de lista de listas.

```
assert eh_triangular_sup([[4,3,1], [0,2,1], [0,0,1]])
assert not eh_triangular_sup([[4,0,0], [0,2,0], [1,0,1]])
```

179. Escreva a função zera_diagonal(M) que zera os valores da diagonal principal da matriz quadrada M. Veja os asserts a seguir:

```
m = [[8, 20, -7],
     [ 5, 1,  3],
     [ 6,  7, 9]]

zera_diagonal(m)
assert  m == [[0, 20, -7], [5, 0, 3], [6, 7, 0]]
```

180. Escreva um programa que multiplique uma matriz MxN por um escalar inteiro K. A matriz é representada na entrada da seguinte forma: a primeira linha da entrada registra os valores de M e N. Nas M linhas seguintes devem ser lidos os N valores de cada linha da matriz. Na última linha é apresentado o escalar pelo qual a matriz deve ser multiplicada. Na saída, você deve imprimir a matriz como ela é representada em python.

Entrada	Saída
3 4 1 2 -1 3 2 1 0 3 4 -1 2 4 5	[[5, 10, -5, 15], [10, 5, 0, 15], [20, -5, 10, 20]]

181. Um jogo implementado em pyhton representa um labirinto através de uma matriz de caracteres com NL linhas e NC colunas. Em cada posição lin, col da matriz, está armazenado um caractere "P" para indicar uma parede, " " (espaço) para representar caminho aberto ou um "*", para indicar a posição em que o jogador está. Pede-se que você crie a função move_direita(labirinto) que move o jogador uma posição à direita da posição em que está, caso o movimento seja possível (o que só é verdade se houver um espaço em branco a sua direita). A função também deve retornar uma tupla contendo a linha e a coluna da posição final do jogador. Veja os asserts.

```
labirinto1 = [
  ['P', '*', ' ', ' '],
```

```
  ['P', ' ', 'P', ' '],
  ['P', 'P', 'P', ' '],
]

assert move_direita(labirinto1) == (0, 2)

assert labirinto1 ==  [
  ['P', ' ', '*', ' '],
  ['P', ' ', 'P', ' '],
  ['P', 'P', 'P', ' '],
]

labirinto2 = [
  ['P', 'P', ' ', ' '],
  ['P', '*', 'P', ' '],
  ['P', 'P', 'P', ' '],
]
assert move_direita(labirinto2) == (1, 1)
```

182. Um jogo implementado em python representa um labirinto através de uma matriz de caracteres com NL linhas e NC colunas. Cada posição lin, col da matriz armazena um caractere "P" para indicar uma parede, " " (espaço) para representar caminho aberto ou um "*" (asterisco) para determinar a posição em que o jogador está. Pede-se que você crie a função movimentos_possiveis(labirinto) que retorna o número de possibilidades de movimento que o jogador tem a partir da posição em que está. Os movimentos podem ser realizados nas seguintes direções: para a direita, para a esquerda, para cima e para baixo. Obviamente, um movimento só é possível nas direções permitidas e caso haja um espaço em branco no local de destino do jogador. Assim, a função só pode retornar 0, 1, 2, 3 ou 4.

```
labirinto1 = [
  ['P', ' ', ' ', ' '],
  ['P', '*', 'P', ' '],
  ['P', 'P', 'P', ' '],
]
assert movimentos_possiveis(labirinto1) == 1
```

ou ainda

```
labirinto2 = [
  ['P', '*', ' ', ' '],
  ['P', ' ', 'P', ' '],
  ['P', 'P', 'P', ' '],
]
assert movimentos_possiveis(labirinto2) == 2
```

183. Escreva uma função que receba duas matrizes e identifique as coincidências entre elas, ou seja, elementos que se encontram na mesma posição (linha e coluna). A função deve retornar uma nova matriz contendo os elementos que coincidem nas suas posições e 0 (zero) nas demais posições. Todas as matrizes recebidas pela função e a matriz retornada devem ter o mesmo número de linhas e colunas. Não haverá matrizes vazias. Veja os asserts:

```
M1 = [[1, 2, 3],
      [4, 5, 6],
      [7, 8, 9]]

M2= [[10, 11, 12],
     [13, 14, 15],
     [ 7,  8,  9]]

M3= [[ 1,  2,  3],
     [13, 14, 15],
     [16, 17, 18]]

assert matriz_coincidencia(M1, M2) == [[0,0,0],[0,0,0],[7,8,9]]
assert matriz_coincidencia(M1, M3) == [[1,2,3],[0,0,0],[0,0,0]]
```

184. Escreva uma função que receba duas matrizes de números inteiros e verifica entre os elementos que se encontram na mesma posição (linha e coluna) quem dos dois é o menor. A função deve retornar uma nova matriz composta pelos menores números encontrados em suas respectivas posições. Todas as matrizes recebidas pela função e a matriz retornada devem ter o mesmo número de linhas e colunas. Não haverá matrizes vazias. Veja os asserts:

```
M1 = [[1,2,3],
```

```
      [13,14,15],
      [7,8,9]]

M2= [[10,11,12],
     [4,5,6],
     [7,8,9]]

M3= [[1,2,3],
     [0,0,0],
     [7,8,9]]

assert matriz_menor(M1, M2) == [[1,2,3],[4,5,6],[7,8,9]]
assert matriz_menor(M1, M3) == [[1,2,3],[0,0,0],[7,8,9]]
```

185. Um pixel é o menor ponto que forma uma imagem digital. Tipicamente, aplicações gráficas representam cada pixel por uma tupla de 3 inteiros com valores entre 0 e 255, que representam os níveis de intensidade de cada uma das componentes de cor: vermelho, verde e azul. Assim, um pixel de vermelho intenso, por exemplo, tem o valor (255, 0, 0). Um pixel amarelo, por outro lado, pode ser criado, "misturando" bastante intensidade de vermelho e verde e pouca intensidade de azul. Por exemplo, pixels de valor (255, 255, 0) representam um amarelo intenso. Se reduzirmos a intensidade de verde desse pixel amarelo, a cor progride para um tom alaranjado - o pixel de (255, 144, 0) é laranja, por exemplo. Pixels brancos, cinzas e pretos têm uma característica em comum: as intensidades das três cores são iguais. Branco e preto, por exemplo, são os pixels (255, 255, 255) e (0, 0, 0), respectivamente. E qualquer pixel da forma (v, v, v) é, portanto, algum nível de cinza. Pede-se que você escreva a função image2gray(imagem) que recebe uma imagem na forma de uma matriz de pixels (tuplas com três inteiros) e que a converte a imagem para preto e branco. Para converter um pixel, você deve implementar o método da média que consiste simplesmente em alterar as intensidades das três componentes para o valor médio das intensidades originais do pixel. Por exemplo, o pixel vermelho mencionado acima seria convertido para o valor (85, 85, 85), dado que 85 é a média dos valores 255, 0 e 0. Já o laranja seria convertido para o valor (133, 133, 133) porque 133 é a média de 255, 146 e 0 (observe que usamos apenas a parte inteira da média). Veja os asserts para compreender a assinatura e a semântica da função.

```
imagem = [
    [(255,   0,  0), (240,   0,  0), (210,   0, 0)],
    [(  0, 255,  0), (  0, 240,  0), (210, 210, 0)],
    [(110,  80, 40), (100, 240, 40), (  0,   0, 0)],
]
image2gray(imagem)
assert imagem == [
    [( 85,  85, 85), ( 80,  80,  80), ( 70,  70,  70)],
    [( 85,  85, 85), ( 80,  80,  80), (140, 140, 140)],
    [( 76,  76, 76), (126, 126, 126), (  0,   0,   0)],
]
```

186. Escreva a função vetor_por_escalar(vetor, escalar)que multiplica vetor por um número inteiro escalar e que retorna como resultado um novo vetor. Cada vetor é representado por uma lista de inteiros e a multiplicação de um vetor por um escalar representa a multiplicação de cada elemento do vetor por este escalar. Veja os asserts abaixo:

```
vetor_1 = [1, 2, 3]
assert vetor_por_escalar(vetor_1, -1) == [-1, -2, -3]
assert vetor_por_escalar(vetor_1, 2) == [ 2, 4, 6 ]
assert vetor_1 == [1, 2, 3]
```

187. Escreva a função altera_vetor_por_escalar(vetor, escalar) que faça a multiplicação de vetor por um escalar. No lugar de retornar um novo vetor, a função deve alterar a lista vetor com o resultado desta multiplicação. Veja os asserts abaixo para compreender a questão:

```
vetor_1 = [1, 2, 3]
altera_vetor_por_escalar(vetor_1, -1)
assert vetor_1 == [-1, -2, -3]
altera_vetor_por_escalar(vetor_1, 2)
assert vetor_1 == [-2, -4, -6]
```

188. Escreva uma função que receba um dicionário contendo informações sobre o estoque de uma livraria (titulo:quantidade). Sua função deve retornar a quantidade de títulos que estão zerados no estoque. Veja os asserts:

```
livros = { "Metamorfose": 30, "O Principe": 0, "Vigiar e Punir": 0, "Dumbo":
22}
assert ausentes(livros) == 2
```

189. Escreva uma função que receba um dicionário contendo informações sobre o saldo bancário de alguns correntistas (nome:saldo). Sua função deve retornar a quantidade de correntistas que estão no vermelho, ou seja, com saldo devedor menor que zero. Veja os asserts:

```
contas = { 'Ana':1000, 'Antonio':-500, 'William':0, 'Carlos':2500, 'Kate':-1300
}
assert devedores(contas) == 2
```

190. Escreva um programa que identifique a letra com maior número de ocorrências em um dado texto. O programa deve ler da entrada padrão o texto e imprimir na saída apenas a letra (minúscula) que mais ocorre, seguida do número de ocorrências. Se houver empate no número de ocorrências de duas ou mais letras, imprima qualquer uma delas. Assuma que o texto não tem caracteres acentuados e/ou especiais. Você não deve usar os métodos count() ou find(). Veja exemplo de entrada e saída.

```
Entrada
Exemplo de entrada e saida.

Saída
e 5
```

191. Codifique em python o algoritmo de ordenação bolha abaixo especificado em pseudo-código. Empacote o algoritmo em uma função chamada bubblesort(dados) que altere a estrutura de dados originalmente passada...

```
função bubble-sort
entrada:
    - dados, uma lista de elementos ordenáveis

    loop:
        swapped = falso
        para cada i de 0 a N - 1:
            se dados[i] > dados[i+1]:
                troca valores de dados[i] e dados[i+1]
                swapped = True

        se nao swapped: leave
```

192. Um aplicativo representa uma agenda de compromissos mensais através de uma matriz de strings. Cada coluna da matriz representa um dia do mês (dia 1, índice 0, dia 31, índice 30). Cada linha da matriz representa uma das horas do dia, das 08h (índice 0) até as 20h (índice 12). Pede-se que você escreva a função: registra_compromisso(agenda, dia, h_inicial, h_final, texto). A função deve alterar a agenda dada em todos o período compreendido entre a hora inicial (h_inicial) e a hora final (h_final) do dia indicado do mês. Para alterar a agenda, o novo compromisso deve sobreescrever qualquer outro que já tiver sido agendado.

```
agenda = [31*[""] for i in range(13)]
registra_compromisso(agenda, 1, 8, 9,"prog1")
assert agenda[0][0] == "prog1"
assert agenda[1][0] == "prog1"
```

193. Escreva a função busca_matriz(m, e) que busca o elemento e na matriz m (representada na forma de lista de listas) e que retorna os índices do elemento e ou None, caso o elemento não exista.

```
matriz = [
    [2, 3, 5, 3, 1],
    [3, 2, 1, 5, 6],
    [1, 2, 3, 2, 1],
]
assert busca_matriz(matriz, 4) == None
assert busca_matriz(matriz, 3) == (0,1)
assert busca_matriz(matriz, 1) == (0,4)
```

194. Escreva a função busca_todos_por_coluna_em_matriz(m, e) que busca o elemento e na matriz m (representada na forma de lista de listas) e que retorna uma lista com todos os índices das posições de ocorrência do elemento. Uma lista vazia é retornada caso o elemento não exista. A ordem de ocorrência dos elementos na lista de saída deve considerar o caminhamento por colunas. Veja os asserts.

```
matriz = [
    [2, 3, 5, 3, 1],
    [3, 2, 1, 5, 6],
    [3, 2, 3, 2, 1],
```

```
]
assert busca_todos_por_coluna_em_matriz(matriz, 4) == []
assert busca_todos_por_coluna_em_matriz(matriz, 3) == [(1,0), (2,0), (0,1),
(2,2), (0,3)]
assert busca_todos_por_coluna_em_matriz(matriz, 1) == [(1,2), (0,4), (2,4)]
```

195. Foram registrados índices pluviométricos de um conjunto de cidades em uma tabela, na qual as linhas representam as cidades e as colunas os dias em que a medição foi realizada. Pede-se que você crie uma função acima_media_plu(tabela, media) que retorne o primeiro valor acima da média histórica encontrado na tabela. O primeiro argumento da função é uma matriz contendo as medições registradas em postos pluviométricos. O segundo argumento é um número real que representa a média histórica do estado. Caso haja mais de uma medida que ultrapasse a média histórica, é suficiente retornar apenas uma. Se não houver nenhum, retorne -1. Veja os asserts abaixo.

```
tabela = [[53.8, 9.7, 29.4, 100.6, 229.2, 128.0],
          [36.4, 32.7, 47.3, 71.2, 111.0, 196.4],
          [17.4, 23.0, 17.9, 29.4, 118.2, 68.4]]
assert acima_media_plu(tabela, 200.0) == 229.2
```

196. Foram registrados índices pluviométricos de um conjunto de cidades em uma tabela, na qual as linhas representam as cidades e as colunas os dias em que a medição foi realizada. Pede-se que você crie uma função maior_indice_plu(tabela, cidades) que retorne a cidade com o maior indice pluviometrico da tabela e o valor deste. O primeiro argumento da função é uma matriz nxm contendo as medições dos pluviômetros registradas em cada cidade. Cada linha, representa uma cidade e as colunas os dias em que as medidas foram tomadas. O segundo argumento é a lista cidades, como o nome já diz, contém o nome de n cidades em que foram tomadas as medidas na ordem das linhas da matriz. Veja os asserts para entender melhor como a função trabalha.

```
tabela = [[27.2, 31.1, 35.7, 85.5, 110.2],
          [4.2, 11.2, 23.7, 66.3, 134.1],
          [28.3, 26.8, 39.6, 69.4, 96.8],
          [9.3, 6.7, 20.6, 40.5, 75.4]]

cidades = ["Camaratuba", "Espinharas", "Gramame", "Jacu"]

assert maior_indice_plu(tabela, cidades) == "Espinharas 134.1"
```

197. Escreva a função eh_escalonada(M) que determina se a matriz M está ou não na forma escalonada completa por linhas. Por definição, uma matriz está na forma escalonada se todos os elementos das posições da diagonal principal da matriz são iguais a 1 e se os demais são iguais a 0 (considerando apenas a porção MxM da matriz ampliada).

```
assert eh_escalonada([[1,2,3], [3,4,5]]) == False
assert eh_escalonada([[1,0,3], [0,1,6]]) == True
```

198. O SPLab, um grande laboratório de pesquisa em engenharia de software, resolveu abrir uma franquia de restaurantes de massas italianas, o SPLabeto. No SPLabeto, cada cliente pode escolher um tipo de massa entre 8 opções, 2 molhos entre 4 tipos diferentes de molhos e 4 acompanhamentos dentre 16 variedades disponíveis. Os clientes podem repetir os mesmos acompanhamentos e molhos, mas tais repetições são contadas nos limites indicados anteriormente.

Para controlar o sistema, um software foi desenvolvido e cada massa, molho e acompanhamento eram convertidos numa lista de códigos que representavam as opções de cada usuário. Neste sistema, as massas eram representados por valores de 0 até 7. Já os molhos iam de 100 a 103 e os acompanhamentos de 1000 até 1015.

Entretanto, o sistema desenvolvido não levava em consideração que o cozinheiro poderia colocar os itens em qualquer ordem. Ou seja, era possível encontrar pedidos que começassem com a massa, depois os acompanhamentos e por fim, os molhos. Pior ainda: o sistema permitia que o cozinheiro inserisse pedidos inválidos, seja por: faltar a massa ou um dos molhos ou um dos acompanhamentos ou por colocar alguma destas opções a mais.

Pede-se que você construa uma função **ajeita_pedido(lista_de_codigos)** que recebe uma lista com os códigos utilizados para um determinado pedido e retorne a lista ordenada destes códigos, caso seja um pedido válido, ou retorne uma lista vazia, caso o pedido seja inválido. Veja os asserts.

```
prato1 = [0, 100, 101, 1000, 1001, 1002, 1000]
prato2 = [1015, 1014, 1013, 1012, 103, 102, 7]
```

```
prato3 = [8, 102, 102, 1002, 1004, 1001, 1000]
prato4 = [1, 1, 1, 1]

assert ajeita_pedido(prato1) == [0, 100, 101, 1000, 1000, 1001, 1002]
assert ajeita_pedido(prato2) == [7, 102, 103, 1012, 1013, 1014, 1015]
assert ajeita_pedido(prato3) == []
assert ajeita_pedido(prato4) == []
```

199. Escreva uma função que multiplique uma matriz A (m x n) por uma matriz B (n x p), representadas como listas de listas de inteiros. A função não deve alterar as matrizes recebidas.

```
mA = [[0, 1, 2], [1, 1, 3]]
mB = [[2, 1], [2, 2], [0, 3]]
assert mult_matrizes(mA, mB) == [[2, 8], [4, 12]]
```

200. As notas dos minitestes práticos das turmas de LP1 estão registradas em uma tabela. As colunas representam cada um dos minitestes e as linhas os alunos. Pede-se que você crie uma função media_alunos(tabela) que recebe como argumento uma matriz contendo as notas dos alunos em cada miniteste. A professora combinou com os alunos que eliminaria a maior e a menor nota dos minitestes de cada aluno. Então você deve calcular a média considerando este requisito: ignorando a menor e a maior nota obtida por cada aluno. A função deve retornar uma lista com as médias de cada aluno (na ordem em que eles aparecem nas linhas da matriz) formatadas com uma casa decimal. Assuma que haverá pelo menos 3 minitestes. Observe os asserts para entender como a função trabalha.

```
tabela_notas = [[10.0, 4.0, 5.0, 0.0],
                [8.0, 7.0, 5.0, 3.0],
                [5.0, 5.0 ,5.0, 5.0],
                [2.0, 10.0, 9.0, 0.0]]

assert media_alunos(tabela_notas) == ['4.5', '6.0', '5.0', '5.5']
```

201. As notas dos minitestes práticos das turmas de LP1 estão registradas em uma tabela. As colunas representam cada um dos minitestes e as linhas os alunos. Pede-se que você crie uma função media_mtps(tabela) que recebe como argumento uma matriz contendo as notas dos alunos em cada miniteste. A função deve calcular a nota média da turma para cada mtp. Ela deve retornar uma lista com as médias de cada miniteste formatadas com uma casa decimal. Observe os asserts para entender como a função trabalha.

```
tabela_notas = [[10.0, 4.0, 5.0, 0.0],
                [8.0, 7.0, 5.0, 3.0],
                [5.0, 5.0 ,5.0, 5.0],
                [2.0, 10.0, 9.0, 0.0]]

assert media_mtps(tabela_notas) == ['6.2', '6.5', '6.0', '2.0']
```

202. Escreva uma função que verifique o resultado de um jogo de bingo. A função resultado_bingo(cartela,numeros) recebe dois parâmetros: cartela e numeros. O primeiro, que representa a cartela do jogador (veja o exemplo da figura abaixo), é uma matriz N x 5 (com N variando de 3 a 5). O segundo é uma lista que representa os números que já foram sorteados. Os números podem variar de 0 a 90. Para considerar uma cartela vencedora, é preciso que todos os números de uma coluna tenham sido sorteados. Sua função deve retornar os números das colunas, caso todos os números destas estejam na lista de números sorteados. Caso a cartela não seja vencedora, "-" deve ser retornado. Veja os asserts abaixo:

```
numeros = [69,73,64,5,55,22,24,88,67,17,55,31,12,16,18,35,47,59,8,10]
cartela_jose = [[5,20,33,55,70], [12,23,40,59,84], [17,28,47,64,88]]
cartela_maria = [[7,25,44,57,62], [15,22,40,50,70], [2,28,37,55,68]]

assert resultado_bingo(cartela_jose, numeros) == "14"
assert resultado_bingo(cartela_maria, numeros) == "-"
```

203. O mês de pagamento do IPVA, na Paraíba, é dado pelo último dígito da placa dos veículos. A tabela a seguir define até quando deve ser o pagamento com desconto de 10% para cota única:

Final de Placa	Mês de Vencimento
1 e 2	janeiro
3 e 4	fevereiro
5	marco
6	abril
7	maio

8	junho
9	julho
0	agosto

Você deve criar um programa que recebe na entrada uma sequencia de placas de veículos (as placas são compostas por LLLDDDD, onde L: letra e D: dígito) e imprime cada placa juntamente com o mês de vencimento do pagamento do IPVA com desconto. Veja um exemplo abaixo:

Entrada	Saída
MXV1234 MOW4321 MMV8062	MXV1234: fevereiro MOW4321: janeiro MMV8062: janeiro

204. Um sistema de controle acadêmico da UFCG usa um mapa de matrículas para médias para cada turma. Pede-se que você escreva a função aprovados(notas_turma) que retorna uma lista das matrículas dos alunos aprovados da turma (alunos com médias maiores ou iguais a 5,0). Veja o assert abaixo como exemplo do funcionamento da função.

```
notas_turma = {
    '112352617': 4.0,
    '112352415': 6.0,
    '112352622': 7.0,
    '112351824': 3.5,
    '112351211': 8.0,
}
assert sorted(aprovados(notas_turma)) == [
    '112351211', '112352415', '112352622',
]
```

205. Os primeiros 3 digitos da matrícula de uma universidade indicam o período de ingresso do estudante. Pede-se que você escreva uma função que dados os estudantes matriculados em uma turma, contabilize quantos estudantes há de cada período. Escreva a função agrupa_por_periodo(alunos) que recebe uma lista de matrículas de estudantes matriculados em uma turma e que retorna um mapa cujas chaves são strings correspondendo aos períodos e cujos valores são o total de estudantes daquele período matriculados. Veja o exemplo de asserts abaixo.

```
turma = [
    '0511114', '0521137', '0611001',
    '0611003', '0611004', '0621006',
    '0811007', '0811009', '0811502',
    '0811604', '0811605',
]
assert agrupa_por_periodo(turma) == {
    '051': 1,
    '052': 1,
    '061': 3,
    '062': 1,
    '081': 5,
}
```

206. Escreva uma função que agrupa em negativos e não-negativos, os números passados como parâmetros numa lista. A função deve retornar um dicionário contendo os números agrupados. A função deve funcionar segundo os asserts abaixo:

```
assert agrupa_negativos([10, -2, -7, 8]) == {"nao-negativos":[10, 8],
"negativos":[-2,-7]}
assert agrupa_negativos([-1, -5]) == {"nao-negativos":[ ], "negativos":[-1, -
5]}
```

207. Escreva uma função que decodifica uma mensagem cifrada. A função **decifra(chave, mensagem)** recebe um dicionário contendo a chave de transformação e a mensagem. Para decodificar a mensagem substitua os caracteres (chave) que aparecem na mensagem pelo correspondente (valor) no dicionário. Sua função deve retornar uma string contendo a mensagem decodificada. Veja os asserts:

```
chave1={'@':'V','a':'v','n':'o','l':'i','#':' ','4':'a','+':'u'}
assert decifra( chave1, '+a4') == 'uva'
assert decifra( chave1, '@nan#al+#4#+a4') == 'Vovo viu a uva'
```

```
chave2 = {'a': 'l', '@': 'a', '#': ' ', 'b': 'o', '+': 'd', 'm': 'u', 'c':
's','n': 'S',



'1': 't', '9': 'e', '3': 'm', 'r': 'f', '4': 'r', 'w': '!','v': 'v', 'y':
'c','x': 'p',



'[': 'g', 'z': 'n', ']': 'i', 'k': ',' ,'7':'P', '5':'b'}




decifra(chave2,'n9#vby9#ybzc9[m]m#a94#9c1@#39zc@[93k#@[4@+9y@#@#m3#x4br9ccb4w')

decifra(chave2,'7@4@59zck#x4br9ccb49cw')
```

208. Escreva a função **agrupa_resumos_iguais()** que recebe uma lista de números inteiros. O resumo de um número é o resultado da soma seus algarismos. A função deve retornar um dicionário cujas chaves são os resumos dos números e o valor de cada resumo é a lista dos números que produziram aquele resumo. Veja os asserts:

```
lista1 = [60, 343, 19, 1230, 51, 123]
assert agrupa_resumos_iguais(lista1) == {6:[60, 1230, 51, 123], 10:[343,19]}
```

209. Sempre que juntamos uma certa quantidade de coisas, surge a necessidade de agrupá-las de acordo com algum critério. Por exemplo, na universidade, os alunos poderiam ser organizados de acordo com o curso; num curso de graduação, os alunos poderiam ser organizados de acordo com os períodos. Foi pensando dessa forma que os cientistas classificaram os elementos químicos para facilitar o estudo da Química. A tabela a seguir lista alguns elementos da tabela periódica juntamente com suas respectivas massas atômicas (massa de um átomo) aproximadas:

Nome	**Massa**	**Símbolo**
Hidrogênio	1	H
Enxofre	32	S
Oxigênio	16	O
Carbono	12	C
Cálcio	40	Ca
Sódio	23	Na
Fósforo	31	P

Uma das tarefas mais básicas num laboratório é o cálculo da massa de uma molécula (agrupamento de átomos). Por exemplo: para calcular a massa molecular da água (H2O), que possui dois átomos de Hidrogênio e um de Oxigênio, fazemos H * 2 + O, onde a massa de cada átomo pode ser obtida na tabela acima. Daí, H * 2 + O é igual a 1 * 2 + 16, que resulta em 18 unidades de massa atômica.

Um cientista precisa de uma maneira automática para calcular a massa de várias moléculas com o intuito de agilizar suas pesquisas. Para ajudá-lo, você precisa desenvolver um programa que realize esse cálculo. O seu programa deve usar os dados da tabela acima.

Entrada

Cada linha da entrada corresponde a um caso de teste diferente. Cada caso de teste é composto por uma sequência de símbolos de elementos químicos seguidos ou não de um inteiro indicando a quantidade de átomos daquele elemento, todos separados por um espaço em branco.

Saída

Para cada caso de teste da entrada o seu programa deve imprimir um número inteiro que indica a massa molecular calculada. O programa deve parar quando a palavra fim for digitada. Dica: para saber se o conteúdo de uma string pode ser convertido para inteiro use a função isdigit().

Entrada	Saída
C O 2 H 2 O C 12 H 22 O 11 fim	44 18 342

210. Escreva uma função que agrupa em pares e ímpares, os números passados como parâmetros numa lista. A função deve retornar um dicionário contendo os números agrupados. A função deve funcionar segundo os asserts abaixo:

```
assert agrupa_pares_impares([10, 24, 97, 88]) == {"pares":[10, 24, 88],
"impares":[97]}
assert agrupa_pares_impares([11, 23, 35]) == {"pares":[ ], "impares":[11, 23,
35]}
```

211. O processo de verificação de contas em uma agência bancária consiste em verificar se o número de conta corresponde ao dígito verificador. O dígito verificador de uma conta é sempre dado pelo resto da divisão do número por 11. Escreva a função **cria_mapa(lista)** que recebe uma lista de contas e retorna um dicionário que tem como chave o dígito verificador das contas. Veja o assert:

```
contas = [5521, 5243, 5680, 4209, 1499]
assert cria_mapa(contas) == {3: [1499], 4: [5680], 7: [5243, 4209], 10: [5521]}
```

212. Os alunos de Programação I do semestre 2012.2 foram divididos em 3 turmas teóricas e 5 turmas práticas. A alocação dos alunos nas turmas teórica e prática é definida em um dicionário. Cada chave do dicionário representa um aluno matriculado. O valor associado à chave é uma tupla em que o primeiro elemento indica a turma teórica e o segundo elemento indica a turma prática. Escreva a função lista_presenca(alocacao, tipo, turma) que receba um dicionário com a alocação dos alunos, uma string que indica se a turma é prática ou teórica ("t" para teórica e "p" para prática) e o número de uma turma e que retorne uma lista com os nomes dos alunos pertencentes a turma especificada.

```
alocacao = {
          "Joao": (2, 4),
          "Maria": (1, 3),
          "Melinda": (3, 5),
          "David": (2, 2),
          "Pedro": (2, 4),
}

assert set(lista_presenca(alocacao, "t", 2)) == set(["Joao", "David", "Pedro"])
assert set(lista_presenca(alocacao, "p", 5)) == set(["Melinda"])
```

213. Os alunos de Programação I do semestre 2012.2 foram divididos em 3 turmas teóricas ("t1", "t2" e "t3"). A alocação dos alunos nas turmas é definida em um dicionário. Cada chave do dicionário representa um aluno matriculado. O valor associado à chave indica a turma. Escreva a função turma_mais_alunos(alocacao) que receba um dicionário com a alocação dos alunos e que retorne o nome da turma que possui mais alunos. Considere que sempre haverá uma única turma com mais alunos do que as outras.

```
alocacao = {
          "Joao": "t2",
          "Maria": "t1",
          "Melinda": "t3",
          "David": "t2",
          "Pedro": "t2",
}

assert turma_mais_alunos(alocacao) == "t2"
```

214. O jogo de Sudoku espalhou-se rapidamente por todo o mundo, tornando-se hoje o passatempo mais popular em todo o planeta. Muitas pessoas, entretanto, preenchem a matriz de forma incorreta, desrespeitando as restrições do jogo. Sua tarefa neste problema é escrever um programa que verifica se uma matriz preenchida é ou não uma solução para o problema. A matriz do jogo é uma matriz de inteiros 9 x 9 . Para ser uma solução do problema, cada linha e coluna deve conter todos os números de 1 a 9. Além disso, se dividirmos a matriz em 9 regiões 3 x 3, cada uma destas regiões também deve conter os números de 1 a 9. O exemplo abaixo mostra uma matriz que é uma solução do problema.

```
1 3 2 | 5 7 9 | 4 6 8
```

```
4 9 8 | 2 6 1 | 3 7 5
7 5 6 | 3 8 4 | 2 1 9
------+-------+------
6 4 3 | 1 5 8 | 7 9 2
5 2 1 | 7 9 3 | 8 4 6
9 8 7 | 4 2 6 | 5 3 1
------+-------+------
2 1 4 | 9 3 5 | 6 8 7
3 6 5 | 8 1 7 | 9 2 4
8 7 9 | 6 4 2 | 1 5 3
```

Entrada

São dadas várias instâncias. O primeiro dado é o número n > 0 de matrizes na entrada. Nas linhas seguintes são dadas as n matrizes. Cada matriz é dada em 9 linhas, em que cada linha contém 9 números inteiros.

Saída

Para cada instância seu programa deverá imprimir uma linha dizendo Instancia k, onde k é o número da instância atual. Na segunda linha, seu programa deverá imprimir SIM se a matriz for a solução de um problema de Sudoku, e NAO caso contrário. Imprima uma linha em branco após cada instância.

Entrada

```
2
1 3 2 5 7 9 4 6 8
4 9 8 2 6 1 3 7 5
7 5 6 3 8 4 2 1 9
6 4 3 1 5 8 7 9 2
5 2 1 7 9 3 8 4 6
9 8 7 4 2 6 5 3 1
2 1 4 9 3 5 6 8 7
3 6 5 8 1 7 9 2 4
8 7 9 6 4 2 1 5 3

1 3 2 5 7 9 4 6 8
4 9 8 2 6 1 3 7 5
7 5 6 3 8 4 2 1 9
6 4 3 1 5 8 7 9 2
5 2 1 7 9 3 8 4 6
9 8 7 4 2 6 5 3 1
2 1 4 9 3 5 6 8 7
3 6 5 8 1 7 9 2 4
8 7 9 6 4 2 1 3 5
```

Saída

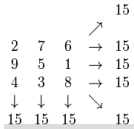
```
Instancia 1
SIM

Instancia 2
NAO
```

(Problema SUDOIME do SPOJ Brasil)

215. Escreva uma função que receba uma matriz e verifica se ela representa um quadrado mágico retornando True ou False (booleano). No quadrado mágico o resultado da soma dos elementos de suas linhas, suas colunas e diagonais

tem o mesmo valor. Além disso, não pode haver elementos repetidos no quadrado. Veja os asserts.




```
quadrado1 = [[2,7,6],[9,5,1],[4,3,8]]
quadrado2 = [[1,2,3],[4,5,6]]
assert eh_quadrado_magico(quadrado1)
assert eh_quadrado_magico(quadrado2)
```

216. Dada uma tabela com os equipamentos elétricos de uma residência, a quantidade de cada equipamento, o tempo de uso mensal em horas e a potência de cada um, você deve escrever uma função calcula_conta(tabela) que calcula a conta de energia. Para saber o consumo de cada equipamento basta multiplicar a quantidade dos equipamentos pela quantidade de horas e depois pela potência. Para indicar o valor da conta considere que um kWh custa R$ 0,28. Veja o exemplo abaixo:

```
tabela = [["Equipamento", "Quantidade", "Tempo de Uso (horas)", "Potencia
(Watts)"],
          ["AR-CONDICIONADO",        1,                    240,
2000],
          ["COMPUTADOR",          2,                    150,
180],
           ["TV",                     3,                    150,
110]]

assert calcula_conta(tabela) == "R$ 163.38"
```

217. A principal unidade de comprimento é o metro. Entretanto, existem situações em que essa unidade deixa de ser prática. Por exemplo, se queremos medir a distância entre João Pessoa e Campina Grande, a unidade metro é muito pequena. Por outro lado, se queremos medir o comprimento de uma formiga, a unidade metro é muito grande. Nesses casos, para facilitar, utilizamos os múltiplos e submúltiplos das unidades de medidas. Os múltiplos e submúltiplos do metro, no Sistema Internacional de Medidas (SI), são apresentados na tabela a seguir:

Fator	Símbolo	Nome
1000	km	quilômetro
100	hm	hectômetro
10	dam	decâmetro
1	m	metro
0.1	dm	decimetro
0.01	cm	centimetro
0.001	mm	milimetro

Um cientista precisa fazer várias medições e somá-las duas a duas. Para ajudá-lo, você precisa desenvolver um programa que realize essa soma para o cientista. O seu programa deve usar os símbolos da tabela acima para realizar a soma de um conjunto de pares de valores medidos pelo cientista. O resultado da soma deve ser impresso na unidade de medida básica (o metro).

Entrada

Cada linha da entrada corresponde a um caso de teste diferente. Cada caso de teste é composto por quatro informações: um inteiro X e sua unidade de medida e mais um inteiro Y e sua unidade de medida, todos separados por um espaço em branco.

Saída

Para cada caso de teste da entrada o seu programa deve imprimir um número real Z que indique a soma X + Y juntamente com sua unidade de medida. O resultado de cada soma deve possuir duas casas decimas e estar na unidade de medida básica.O programa deve parar quando ambos os valores a somar forem iguais a zero.

Entrada	Saída
20 cm 1 m 2 km 80 cm 0 cm 0 m	1.20 m 2000.80 m

218. Escreva a função zera_negs(M) que zera todos os valores negativos da matriz M.

```
m = [[10, 20, -7],
     [ 5, 14,  3],
     [ 6,  7, -2]]

zera_negs(m)
assert  m == [[10, 20, 0], [5, 14, 3], [6, 7, 0]]
```

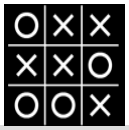
219. Escreva a função procura_elemento() que verifica se existe um dado elemento em uma matriz. A função recebe a matriz e o elemento e retorna uma tupla (linha, coluna) com a posição do elemento na matriz. Caso o elemento não exista, a função deve retornar (-1,-1). Assuma que todos os elementos da matriz são diferentes.

```
m = [[1, 2],[ 3, 4], [5,6]]
assert procura_elemento(m,6) == (2,1)
assert procura_elemento(m,7) == (-1,-1)
```

220. Escreva uma função que produza uma matriz de inteiros a partir de uma lista de inteiros. A função recebe como parâmetros as dimensões M e N da matirz e uma lista de inteiros com exatamente M*N inteiros a serem dispostos na matriz. Os valores são dispostos da primeira linha à última e, em cada linha, da primeira coluna à última. Veja os asserts abaixo.

```
valores = [1,2,3,4,5,6,7,8]
assert constroi_matriz(2, 4, valores) == [[1,2,3,4], [5,6,7,8]]
assert valores == [1,2,3,4,5,6,7,8]
```

221. Escreva uma função que verifica se houve vencedor de um jogo da velha e quem foi. O tabuleiro é representado por uma matriz 3X3 e a função sempre recebe um jogo válido. Os jogadores utilizam os símbolos 'X' e 'O'. Quando há um espaço vazio no tabuleiro ele é representado pelo caracter '-'. O retorno da função deve ser 'X ganhou', se houver uma sequencia de 3 'X' na linha, coluna ou diagonais. O retorno da função deve ser 'O ganhou', se houver uma sequencia de 3 'O' na linha, coluna ou diagonais. O retorno da função deve ser 'Ninguem ganhou', se não houver vencedor. Veja os asserts:



```
tabuleiro1 = [['O', 'O', 'X'],['X', 'O', 'O'],['O', 'O', 'X'] ]
tabuleiro2 = [['O', 'X', 'X'],['X', 'X', 'O'],['O', 'O', 'X'] ]
assert jogo_da_velha(tabuleiro1) == 'O ganhou'
assert jogo_da_velha(tabuleiro2) == 'Ninguem ganhou'
```

222. Sexta-feira é dia de Cantinho Universitário! Descobrimos que um dos alunos de Prog1 é conhecido como "pegador". O nome do mesmo não será revelado para não comprometer sua reputação perante as colegas. Toda sexta-feira ele fica com várias garotas, ou seja, N garotas. Geralmente ele fica com uma, fica com outra, passa um tempo com os colegas conversando, depois fica com outra, e assim por diante. Estamos interessados em saber qual foi o período mais longo em que o "pegador" ficou com uma garota e também o tempo total em que ele estava sem ninguém, apenas conversando com os colegas.

Entrada

A entrada inicia-se com um inteiro N 1, indicando o número de garotas que ficou com o “pegador”. Em seguida, as próximas N linhas são composta por dois inteiros não negativos que indicam, respectivamente, o horário em que o “pegador” começou a ficar com a garota e o horário em que o mesmo a largou.

Saída

Seu programa deve imprimir duas linhas na saída como resposta: a primeira linha deve apresentar o maior período em que o "pegador" ficou com uma garota e a segunda o tempo total em que o "pegador" estava apenas conversando com os colegas. O exemplo de entrada e saída apresentado abaixo deve ser seguido rigorosamente.

Observações

O happy hour no Cantinho Universitário sempre começa quando o "pegador" fica com a primeira garota. O happy hour termina quando ele larga a última garota. Os horários de início e término de uma "pegação" são sempre exatos. Por fim, o happy hour sempre acontece num mesmo dia.

Entrada	Saída
3 3 6 8 15 16 21	7 horas durou a maior pegada 3 horas sem pegar ninguem

223. ROT13 é uma técnica popular para encriptar palavras. Seu funcionamento é simples. Para encriptar uma palavra, cada letra da palavra original deve ser deslocada em 13 posições no alfabeto. Observe o esquema de deslocamento na figura. Escreva a função descloca_palavra(palavra, n) que recebe uma palavra a ser encriptada e um número inteiro n que determina o número de posições para o deslocamento. É importante notar que o deslocamento é circular, ou seja, após a última letra do alfabeto, seguimos para a primeira e recomeça a sequência, por isso trata-se de uma "rotação". Por exemplo: 'A' com um deslocamento 3 transforma-se em 'D'. 'Z' com um deslocamento 1 transforma-se em 'A'. A sua função deve retornar a palavra encriptada. Veja os asserts a seguir:

```
assert desloca_palavra('bala',4) == 'fepe'
assert desloca_palavra('mfmf',9) == 'vovo'
```

224. Uma empresa de segurança patrimonial mantém um registro de senhas e usuários. Em algumas situações, a mesma senha é compartilhada por mais de um usuário. Pede-se que você implemente a função**quantidade_usuarios(cadastro)** que retorna a quantidade de pessoas que tem senha de acesso. O parâmetro cadastro é um dicionário em que as chaves são senhas e os valores são listas de nomes de usuários que compartilham aquela senha. Vale ressaltar que a senha 9999 é reservada para o 'administrador' do sistema. Ele pode aparecer no cadastro, mas não deve ser contabilizado na quantidade de usuários que a função deve retornar. Veja os asserts abaixo:

```
lsd = {1234:['Andrey'], 1226:['Nazareno', 'Livia'], 9999:['administrador'] }
deq = {1114:['Ana'] }

assert quantidade_usuarios(lsd) == 3
assert quantidade_usuarios(deq) == 1
```

225. Os aeroportos são identificados através de um código definido pela IATA (Associação Internacional de Transporte Aéreo). O código IATA é formado por 3 letras e designa unicamente um aeroporto. Por exemplo, o código IATA para o Aeroporto João Suassuna em Campina Grande é CPV, o Aeroporto Tom Jobim no Rio de Janeiro é GIG e o Aeroporto de Guarulhos em São Paulo é GRU.

Uma outra informação importante vinculada a cada aeroporto é a indicação dos vôos diretos que tem origem no referido aeroporto. Por exemplo, os aeroportos que são contemplados com vôos diretos que partem de CPV (aeroporto de Campina Grande) são REC e SSA. Isso indica que é possível sair de CPV e chegar em REC, sem escalas. Da mesma forma, é possível sair de CPV e chegar em SSA, sem escalas.

Um roteiro de viagem aérea pode ser visto como uma composição de vôos diretos entre aeroportos. O roteiro Campina Grande/Recife/São Paulo/Brasília/Campina Grande indica que o vôo sai de campina Grande para Recife, depois de Recife para São Paulo, depois de São Paulo para Brasília e, finalmente, de Brasília para Campina Grande.

Crie uma função que recebe dois mapas (um com o mapeamento do código IATA para os aeroportos e outro com as informações de vôos diretos de cada aeroporto) e um roteiro de viagem aérea. A função deve retornar True se o roteiro é possível de ser realizado e False, caso contrário. Vamos assumir que cada cidade tem apenas um aeroporto e que o roteiro de viagem contempla pelo menos duas cidades. Veja os asserts abaixo:

```
iata = {"Campina Grande": "CPV",
       "Recife": "REC",
       "Salvador": "SSA",
       "Brasilia": "BSB",
       "Sao Paulo": "GRU",
       "Rio de Janeiro": "GIG"}
```

```
voos = {"CPV": ["REC", "SSA"],
       "REC": ["CPV", "BSB", "GRU", "GIG"],
       "SSA": ["REC", "GRU", "GIG"],
       "BSB": ["CPV", "GIG", "GRU"],
       "GRU": ["GIG", "BSB"],
       "GIG": ["GRU", "REC"]}


assert eh_roteiro(iata, voos, "Campina Grande/Recife/Rio de Janeiro")
assert eh_roteiro(iata, voos, "Sao Paulo/Rio de Janeiro/Recife/Brasilia")
assert not eh_roteiro(iata, voos, "Recife/Rio de Janeiro/Salvador/Recife")
```

226. No jogo "zerinho ou um", cada um dos participantes indica ao mesmo tempo com as mãos uma escolha entre 0 e 1. Em seguida, observa-se se alguém escolheu um valor diferente de todos os participantes. O vencedor é exatamente a pessoa que escolheu um valor diferente dos outros. Caso não haja um único valor diferente do restante então não há vencedor.

Entrada

Cada linha da entrada representa uma partida de zerinho ou um. Cada partida, por sua vez, é composta por 0s e 1s que representam as jogadas de cada jogador da partida. Os jogadores são numerados da esquerda para a direita de 1 a N (N sendo o número de jogadores da partida, que pode variar). Assuma que em uma partida haverá no mínimo 3 jogadores. O fim da entrada é demarcado por uma linha contendo apenas o valor -1.

Saída

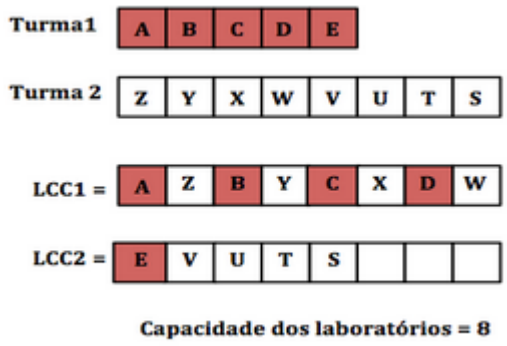
Seu programa deve imprimir na saída o participante vencedor em cada uma das partidas. Caso não haja vencedor, o caractere “X” (maiúsculo) deve ser impresso.

Entrada	Saída
0 0 1 0 1 1 1 0 1 1 -1	3 X 1

227. Escreva a função **troca_chave()** que recebe um dicionário e retorna o dicionário inverso. As chaves do dicionário inverso devem ser os valores do dicionário original e os valores do dicionário inverso devem ser a chave do dicionário original. Considere que os valores do dicionário original serão únicos, para garantir a unicidade das chaves no novo dicionário. Veja os exemplos.

```
assert troca_chave({1:2}) == {2:1}
assert troca_chave({1:2, 2:3, 3:4}) == {2:1, 3:2, 4:3}
assert troca_chave({ '@':'V','a':'v', 'n':'o'}) == { 'V':'@','v':'a', 'o':'n'}
```

228. As provas de Programação 1 são feitas simultaneamente nos laboratórios LCC1 e LCC2. Os professores distribuem os alunos nos laboratórios de modo a preencher primeiro as posições do LCC1. Pede-se que você escreva uma função distribui_alunos(turma1, turma2, capacidade) que distribui os alunos nos dois laboratórios. Os parâmetros turma1 e turma2 são listas com as matriculas dos alunos das turmas. O parâmetro capacidade é um número natural que informa a capacidade máxima de alunos nos laboratórios. A distribuição dos alunos segue a seguinte lógica: você deve intercalar um aluno da primeira turma com outro da segunda, e assim sucessivamente. Os alunos devem ser distribuídos na ordem em que aparecem nas turmas. Note que as turmas devem ter pelo menos um aluno e podem ter tamanhos diferentes. A distribuição deve começar pela turma1. O segundo laboratório só deve ser preenchido após todas as posições do primeiro terem sido ocupadas. A função deve retornar uma lista contendo duas listas com os alunos alocados em cada laboratório. Assuma que a quantidade de alunos não excede a capacidade total dos dois laboratórios.




```
t1 = [10,38,87,22,25]
t2 = [43,21,96,33,85,17,94]
assert distribui_alunos(t1, t2, 6) == [[10, 43, 38, 21, 87, 96],
                                       [22, 33, 25, 85, 17, 94]]
```

229. O Twitter implementa o conceito de seguidor. A idéia é que um usuário pode optar por "seguir" outros usuários, o que significa que ele passará a receber as mensagens enviadas ou “retuitadas” pelos usuários que ele segue. A informação de quem segue quem é mantida no que se chama de matriz de seguidores. Trata-se de uma matriz quadrada de dimensão N, onde N é o número de usuários da rede (usuários são numerados de 0 a N-1). Cada célula da matriz armazena um bit, que registra se o usuário da linha i é seguido ou não pelo usuário da coluna j. Por exemplo, na matriz da figura abaixo, o valor 1 contido na célula na linha 2, coluna 3 indica que o usuário de índice 2 (“carla”) é seguido pelo usuário de índice 3 (“daniel”). Na diagonal principal, a matriz armazena o valor -1, apenas para indicar que o valor é irrelevante, já que cada usuário não pode seguir a si mesmo. Pede-se que você escreva a função alcance_retweet(u, seguidores) que recebe um índice de usuário (u) e a matriz de seguidores (seguidores) e que determina os usuários que devem receber uma mensagem enviada pelo usuário u, caso ela seja “retuitada” por todos os seguidores de u (interessam apenas os “retuítes” dos seguidores diretos de u). A função deve retornar uma lista com N números 0s ou 1s, representando os N usuários, indicando que usuários receberão a mensagem (o próprio usuário u só deve ser incluído na lista resultante, se ele mesmo receber a mensagem “retuitada” por algum de seus próprios seguidores). Um valor 1 na posição i indica que o usuário i receberá a mensagem. Um valor 0, indica que o usuário

		0	1	2	3	4
0	andré	0	-1	0	0	1	0
1	beatriz	1	0	-1	0	1	0
2	carla	2	0	0	-1	1	1
3	daniel	3	1	1	0	-1	0
4	eduardo	4	0	1	1	0	-1

não a receberá. Veja os exemplos dados abaixo.

```
M = [[-1,0,0,1,0], [0,-1,0,1,0], [0,0,-1,1,1], [1,1,0,-1,0], [0,1,1,0,-1]]
assert alcance_retweet(0, M) == [1,1,0,1,0]
assert alcance_retweet(4, M) == [0,1,1,1,1]
```

230. Um jogo implementado em python representa um labirinto através de uma matriz de caracteres com NL linhas e NC colunas. Cada posição lin, col da matriz armazena um caractere "P" para indicar uma parede, " " (espaço) para representar caminho aberto ou um "*" (asterisco) para determinar a posição em que o jogador está. Pede-se que você crie a função movimentos_diagonais(labirinto) que retorna o número de possibilidades de movimento que o jogador tem a partir da posição em que está. Os movimentos podem ser realizados apenas nas 4 diagonais. Obviamente, um movimento só é possível nas direções permitidas e caso haja um espaço em branco no local de destino do jogador.

```
labirinto1 = [
  ['P', ' ', ' ', ' '],
  ['P', '*', 'P', ' '],
  ['P', 'P', 'P', ' '],
]
assert movimentos_diagonais(labirinto1) == 1

labirinto2 = [
  ['P', '*', ' ', ' '],
```

```
  [' ', ' ', ' ', ' '],
  ['P', 'P', 'P', ' '],
]
assert movimentos_diagonais(labirinto2) == 2
```