Iniciado em	quarta, 29 set 2021, 11:23
Estado	Finalizada
Concluída em	domingo, 17 out 2021, 17:29
Tempo empregado	18 dias 6 horas
Avaliar	**10,00** de um máximo de 10,00(**100**%)

Aviso!

?

Questão 1

Correto

Atingiu 1,00 de 1,00

Sozinho

Vítor gosta das coisas bem organizadas. Uma das coisas que ele gosta de manter organizada são suas meias. Elas estão quase totalmente certas, mas ele suspeita que um pé esteja sem o seu par. Ele codificou cada cor de meia com um número, e enviou-os a você para verificar se todos tem um par.

Seu trabalho é criar um programa que diga se todas as meias tem pares ou não, e caso haja alguma meia sozinha, dizer qual a cor dela (o número que representa a cor).

Entrada

A entrada consiste em 2 linhas. Na primeira, um inteiro n onde $1 \leq n \leq 10000$ representando o número de meias.

A segunda linha contém n inteiros separados por espaço. Cada número representa uma cor de meia. É garantido que haverá entre 0 e 1 inteiros que não estão repetidos na sequência. Pode ser que um mesmo número tenha mais de 1 par.

Os números estão **ordenados** em ordem **crescente.**

Saída

Você deve imprimir **tudo certo** se todas as meias tiverem um par. Caso contrário imprima i **sozinho**, onde i é o valor da meia sem um par.

Observações

- No caso de teste 1 todos os números possuem pares. Logo, a saída é **tudo certo**.
- No caso de teste 2, o número 5 é o único que não tem um par, então a saída é **5 sozinho.**

For example:

Input	Result
10 1 1 2 2 4 4 5 5 10 10	tudo certo

Aviso!

Input	Result
9 1 1 2 2 4 4 5 8 8	5 sozinho
7 1 1 1 1 3 4 4	3 sozinho

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
qtd_meias = int(input().strip())
cores = input().split()
cores_diferentes = []
sozinhas = []
for cor in cores:
  if cor not in cores_diferentes:
    cores_diferentes.append(cor)

for cor in cores_diferentes:
  if cores.count(cor) % 2 != 0:
    sozinhas.append(cor)

if len(sozinhas) == 0:
  print("tudo certo")
else:
```

	Input	Expected	Got	
✔	10 1 1 2 2 4 4 5 5 10 10	tudo certo	tudo certo	✔
✔	9 1 1 2 2 4 4 5 8 8	5 sozinho	5 sozinho	✔

Aviso!

?

	Input	Expected	Got	
✔	7 1 1 1 1 3 4 4	3 sozinho	3 sozinho	✔
✔	6 1278 1278 1278 1278 3562 3562 3562 3562 3562 3562	tudo certo	tudo certo	✔
✔	19 400 424 424 424 424 2027 2027 2027 2027 2027 2027 3986 3986 3986 3986 4085 4085 8162 8162	400 sozinho	400 sozinho	✔
✔	25 577 577 577 577 577 577 1853 1853 1853 1853 1853 1853 2632 2632 2632 2632 3877 3877 6044 6044 6044 6044 6044 6798 6798 8630	8630 sozinho	8630 sozinho	✔

Passou em todos os teste! ✔

A primeira coisa a se notar nesse problema é que se n for par não é possível ter um número sozinho, então pode-se retornar **tudo certo** diretamente.

Feito isso, existem algumas opções para se resolver a questão. A primeira e mais óbvia é tirar vantagem da ordenação do vetor, e verificar para cada número se algum dos seus vizinhos tem o mesmo valor que ele. Caso tenha, ele possui um par. Caso **os dois vizinhos** sejam diferentes, esse é o número alvo. Tomar cuidado com os números das pontas que só possuem um vizinho.

Outra alternativa tem a ver com a operação XOR. Ela possui 2 características interessantes:

1. $a \oplus 0 = a$

2. $a \oplus a = 0$

Ou seja, XOR de qualquer valor com 0 é o próprio valor, e XOR de um valor com ele mesmo se "cancela" e vira zero. Note que os números do vetor vem em pares, exceto por 1. Vamos chamar esse número solitário de a_k, e todos os outros de a_i. Podemos fazer: $a_1 \oplus a_2 \oplus a_3 \oplus \ldots \oplus a_k \ldots \oplus a_n = a_k$, pois todos os pares vão se "cancelar" pela propriedade 2 do XOR, e pela propriedade 1, $0 \oplus 0 \oplus 0 \oplus \ldots \oplus a_k = a_k$.

Question author's solution (Python3):

```
n = int(input())
if not n&1:
    print("tudo certo")
else:
```

Aviso!

```
    numbers = input().split()
    acc = 0
    for num in numbers:
        acc = acc ^ int(num)
    print(f'{acc} sozinho')

'''
Solução alternativa (Vanessa)
n = int(input())
numeros = input().split()
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão 2

Correto

Atingiu 1,00 de 1,00

Ordenando Números

Crie um programa que lê 7 números inteiros e os armazena em uma lista $numeros$.

Então imprima a lista $numeros$ ordenada em ordem crescente.

Entrada
A entrada consiste em 7 números inteiros.

Saída
A saída na lista com os números ordenados em ordem crescente.

Observações
- No primeiro exemplo de teste, foi digitado -90, 5, 6, -1, 98, 0 e 10, retornando [-90, -1, 0, 5, 6, 10, 98].
- No segundo exemplo de teste, foi digitado 10, 9, 8, 7, 6, 5 e 4 retornando [4, 5, 6, 7, 8, 9, 10].
- No terceiro exemplo de teste, foi digitado 1, 2, 2, 1, 40,10 e -1, retornando [-1, 1, 1, 2, 2, 10, 40].

For example:

Input	Result
-90 5 6 -1 98	[-90, -1, 0, 5, 6, 10, 98]

Aviso!

?

Input	Result
10 9 8 7 6 5 4	[4, 5, 6, 7, 8, 9, 10]
1 2 2 1 40 10 -1	[-1, 1, 1, 2, 2, 10, 40]

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
QTD_INTEIROS = 7
lista = []

for x in range(QTD_INTEIROS):
  lista.append(int(input()))

lista.sort()

print(lista)

```

Aviso!

?

	Input	Expected	Got	
✔	-90 5 6 -1 98 0 10	[-90, -1, 0, 5, 6, 10, 98]	[-90, -1, 0, 5, 6, 10, 98]	✔
✔	10 9 8 7 6 5 4	[4, 5, 6, 7, 8, 9, 10]	[4, 5, 6, 7, 8, 9, 10]	✔
✔	1 2 2 1 40 10 -1	[-1, 1, 1, 2, 2, 10, 40]	[-1, 1, 1, 2, 2, 10, 40]	✔
✔	9999 -9999 12 546 756 777 444	[-9999, 12, 444, 546, 756, 777, 9999]	[-9999, 12, 444, 546, 756, 777, 9999]	✔
✔	0 0 0 0 0 0 0	[0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0]	✔

Aviso!

?

	Input	Expected	Got	
✔	10393 549859485 4837372 -43898 -2 0 -1	[-43898, -2, -1, 0, 10393, 4837372, 549859485]	[-43898, -2, -1, 0, 10393, 4837372, 549859485]	✔

Passou em todos os teste! ✔

Question author's solution (Python3):

```
1 i = 0
2 numeros = []
3 while i < 7:
4     numeros.append(int(input()))
5     i += 1
6
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão 3

Correto

Atingiu 1,00 de 1,00

História de Pescador

Crie um programa que lê strings atribuindo a variável $peixe$ e as armazena na lista $rede$. As strings devem ser lidas até que seja lido "acabei".

Após isso, deve ser impressa a mensagem: "Hoje eu pesquei:" e os elementos da lista $rede$, um em cada linha.

Entrada
A entrada contém uma quantidade variável de strings.

Saída
A saída consiste em uma mensagem e os elementos da lista lida.

Observações
- No primeiro exemplo de teste, foi digitado "bacalhau" e "acabou", retornando "Hoje eu pesquei: bacalhau".
- No segundo exemplo de teste, foi digitado "salmão", "atum" e "acabou", retornando "Hoje eu pesquei: salmão atum".
- No terceiro exemplo de teste, foi digitado "nada acabou", retornando "Hoje eu pesquei: nada".

For example:

Input	Result
`bacalhau` `acabou`	`Hoje eu pesquei:` `bacalhau`

`salmão`
`atum`
`acabou`
	`Hoje eu pesquei:`
	`salmão`
	`atum`

Aviso!

?

Input	Result
nada acabou	Hoje eu pesquei: nada

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
rede = []
peixe = ''

while peixe != 'acabou':
  peixe = input()
  if peixe != 'acabou':
    rede.append(peixe)

print("Hoje eu pesquei:")
for p in rede:
  print(p)
```

	Input	Expected	Got	
✔	bacalhau acabou	Hoje eu pesquei: bacalhau	Hoje eu pesquei: bacalhau	✔
✔	salmão atum acabou	Hoje eu pesquei: salmão atum	Hoje eu pesquei: salmão atum	✔
✔	nada acabou	Hoje eu pesquei: nada	Hoje eu pesquei: nada	✔

Aviso!

?

	Input	Expected	Got	
✔	peixe-palhaço girafa acabou	Hoje eu pesquei: peixe-palhaço girafa	Hoje eu pesquei: peixe-palhaço girafa	✔
✔	baleia golfinho tubarão acabou	Hoje eu pesquei: baleia golfinho tubarão	Hoje eu pesquei: baleia golfinho tubarão	✔
✔	acabou	Hoje eu pesquei:	Hoje eu pesquei:	✔

Passou em todos os teste! ✔

Question author's solution (Python3):

```
1  rede = []
2  peixe = ""
3
4  while peixe != "acabou":
5      peixe = input()
6      if peixe != "acabou":
7          rede.append(peixe)
8
9  print("Hoje eu pesquei:")
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão **4**

Correto

Atingiu 1,00 de 1,00

Álgebra linear, não. Matemática.

Cedo ou tarde, você irá perceber que o ramo mais útil da matemática é a álgebra linear. Em alguns lugares, nem sequer utilizam o nome álgebra linear. Usam simplesmente matemática! Neste problema, você irá utilizar um pedacinho bem minúsculo da álgebra linear, voltado para geometria.
Na escola, nós aprendemos a determinar se um par de vetores bi ou tridimensionais é ortogonal (ou seja, se o ângulo entre os vetores é de 90^o) calculando seu produto interno euclidiano:

$$u \cdot v = x_u \cdot x_v + y_u \cdot y_v \quad \text{(para 2 dimensões - bidimensionais)}$$

ou

$$u \cdot v = x_u \cdot x_v + y_u \cdot y_v + z_u \cdot z_v \quad \text{(para 3 dimensões - tridimensionais)}$$

Dois vetores u e v são ortogonais se, e somente se, $u \cdot v = 0$. Neste problema, você deve determinar se um dado par de vetores n-dimensionais u e v é ortogonal.

Entrada

A primeira linha da entrada contém um único inteiro n ($1 \leq n \leq 10^5$), que indica a dimensão dos vetores a serem lidos, ou seja, o número de coordenadas cartesianas de um vetor.

A segunda linha contém n inteiros u_i ($-100 \leq u_i \leq 100$) separados por espaço, que são as n coordenadas do vetor u.

A terceira linha contém n inteiros v_i ($-100 \leq v_i \leq 100$) separados por espaço, que são as n coordenadas do vetor v.

Saída

Se os vetores u e v forem ortogonais, imprima a mensagem "ortogonais" (sem aspas duplas). Caso contrário, imprima o valor do produto interno euclidiano entre eles.

Aviso!

Exemplo

?

- Para ler as n coordenadas de um vetor em uma única linha, basta utilizar o seguinte código: vet = list(map(int,input().strip().split()))[:n]
- No primeiro caso de teste, os vetores são ortogonais pois (-3)*1 + (-4)*1 + 2*1 + 5*1 = 0.

For example:

Input	Result
4 -3 -4 2 5 1 1 1 1	ortogonais
3 -2 0 1 1 2 -2	-4
8 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1	ortogonais

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
1
2  from typing import no_type_check_decorator
3
4
5  dimensao = int(input())
6
7  vetor_u = list(map(int, input().strip().split(' ')))
8  vetor_v = list(map(int, input().strip().split(' ')))
9  produto_ortogonal = 0
10
11 for i in range(dimensao):
12   produto_ortogonal += vetor_u[i]*vetor_v[i]
13
14 if produto_ortogonal == 0:
15   print('ortogonais')
```

Aviso!

	Input	Expected	Got	
✔	4 -3 -4 2 5 1 1 1 1	ortogonais	ortogonais	✔
✔	3 -2 0 1 1 2 -2	-4	-4	✔
✔	8 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1	ortogonais	ortogonais	✔
✔	3 2 -2 1 3 2 -2	ortogonais	ortogonais	✔
✔	2 1 7 2 -3	-19	-19	✔
✔	1 1 -2	-2	-2	✔

Passou em todos os teste! ✔

A questão pode ser resolvida, iterando sobre o número n de coordenadas para multiplicar os valores de mesma posição n_i, e somando os resultados. Caso o resultado final da soma seja igual a zero, os vetores são ortogonais.

Question author's solution (Python3):

```
def ortogonais(v1, v2):
    calc = 0
    for i in range(len(v1)):
        calc += v1[i] * v2[i]
    if(calc==0):
```

Aviso!

```
    if(calc==0):
        print("ortogonais")
    else:
        print(calc)

#Numero de elementos que as listas terao
n = int(input())

#Valor dos elementos de cada lista
v1 = list(map(int,input().strip().split()))[:n]
v2 = list(map(int,input().strip().split()))[:n]
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão **5**

Correto

Atingiu 1,00 de 1,00

Elementos repetidos

Escreva um programa que receba como entrada n $1 \geq n \geq 100$ elementos de uma lista, e como saída, imprima **True** caso exista algum elemento que apareça mais de uma vez, e **False** caso contrário.

Entrada
Uma lista de números.

Saída
True ou False dependendo da existência ou não de elementos repetidos.

For example:

Input	Result
1 2 3 4 5 6 7 8 9	False
1 1 2 3 4 5 6 7 8	True
1 1 1 1 1 2	True

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
lista = input().strip().split(' ')


def existeRepetido(lista):
    lista_unica = []
    for x in lista:
        if x not in lista_unica:
```

Aviso!

```
8         lista_unica.append(x)
9     for x in lista_unica:
10      if lista.count(x) > 1:
11        return True
12    return False
13
14  print(existeRepetido(lista))
```

	Input	Expected	Got	
✔	1 2 3 4 5 6 7 8 9	False	False	✔
✔	1 1 2 3 4 5 6 7 8	True	True	✔
✔	1 1 1 1 1 2	True	True	✔
✔	1	False	False	✔
✔	0 1 2 3 4 5 6 7	False	False	✔
✔	1 5 4 7 8 2 -2	False	False	✔

Passou em todos os teste! ✔

Uma forma de resolver a questão é iterando sobre a lista e para cada numero i da lista realizar uma iteração iniciando em i+1 para comparar os números a frente com o número i. Caso encontre um valor repetido, para a execução da iteração e retorna True.

Question author's solution (Python3):

```
1  def elementos_duplicados(l):
       for i in range(len(l)):
           for j in range(i+1,len(l)):
```

Aviso!

```
            if l[i] == l[j]:
                return True
    return False


l = [int(i) for i in input().split()]
print (elementos_duplicados(l))

# Solucao alternativa: Vinicius
#flag = [0]*101

#lista = [int(x) for x in input().split()]
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão **6**

Correto

Atingiu 1,00 de 1,00

Bob e seus datagramas

Alice e Bob são personagens lendários usados para apresentar de maneira didática situações onde há uma comunicação entre duas entidades *A* e *B*.

Imagine que Bob deseja enviar uma imagem para Alice. Na Internet, isto é feito quebrando a imagem em vários pedaços, chamados de datagramas, e enviando um datagrama de cada vez. O problema é que os datagramas podem se espalhar por diferentes caminhos para chegarem ao seu destino. Deste modo, os datagramas podem chegar desordenados ao computador de Alice. Para evitar esta confusão, os datagramas são enviados por Bob com um número de sequência. Bob sempre envia um datagrama com um número de sequência maior que o número de sequência do datagrama que foi enviado por último.

Desta maneira, Alice pode reconstruir a imagem simplesmente ordenando os datagramas que ela recebeu, pelos respectivos números de sequência. Mas antes de fazer isto, Alice te pediu ajuda para calcular um número.

Ela quer saber quantas inversões aconteceram no envio da imagem. O número de inversões é o número de pares $\{S_i, S_j\}$ tais que $i < j$ e $S_i > S_j$ onde S_1 é o número de sequência do datagrama que chegou primeiro, S_2 é o número de sequência do datagrama que chegou em segundo lugar, S_3 é o número de sequência do datagrama que chegou em terceiro lugar, e assim por diante.

Entrada

A primeira e única linha contém uma sequência de N ($1 \leq N \leq 10^3$) inteiros não-negativos $S_i (1 \leq S_i \leq 10^9)$ que são os números de sequência dos datagramas em ordem de chegada ao computador de Alice.

Saída

Imprima uma linha com o número de inversões, como descrito no enunciado (o número de pares tais que $i < j$ e $S_i > S_j$).

Observações

- No segundo caso teste, o número 5 gera situações de inversão com os números 2, 3, e 4. Portanto o número de inversões é 3.
- No terceiro caso de teste, o número 5 gera situações de inversão com os números 2, 4 e 3 e o número 4 gera inversão com o número 4. Portanto o número de inversões é 4.

For example:

Aviso!	Result

Input	Result
1 4 5 9 15	0
1 5 2 3 4	3
1 5 2 4 3	4

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
sequencia = list(map(int, input().strip().split(' ')))

inversoes = 0

for x in sequencia:
  for i in range(sequencia.index(x), len(sequencia)):
    if x > sequencia[i]:
      inversoes += 1

print(inversoes)
```

	Input	Expected	Got	
✔	1 4 5 9 15	0	0	✔
✔	1 5 2 3 4	3	3	✔
✔	1 5 2 4 3	4	4	✔

Aviso!

?

	Input	Expected	Got	
✔	5 4 3 2 1	10	10	✔
✔	2239 306784401 9149179 0 884939 1000000000 740 54038 9983 1 63888 9698 2 67963 78 37207 921339 948359 55 279528 4 341002 823 21 1299 185735498 824948933 93255364 22 607066 834148 75193839 994184	230	230	✔
✔	6940 84092800 762 3837 217 69998638 38821 886 5 555693504 977 993807865 9579134 7 78 952311 56 1000000000 781793032 1237847 769272 55355425 4405 9234062 6714 36 41 56769414 916 897001113 0 919408 1773	272	272	✔

Passou em todos os teste! ✔

A questão pode ser resolvida, capturando todos os números e iterando sobre eles. A cada interação i, avalia-se o número da posição i em relação a todos os outros que já passaram. Se algum número que já passou é maior do que o número na posição i, então contabiliza-se uma inversão.

Question author's solution (Python3):

```
def datagrama(l):
    inversoes = 0
    for i in range(len(l)):
        for j in range(i):
            if(l[j] > l[i]):
                inversoes += 1

    print(inversoes)
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão 7

Correto

Atingiu 1,00 de 1,00

Fibonacci III

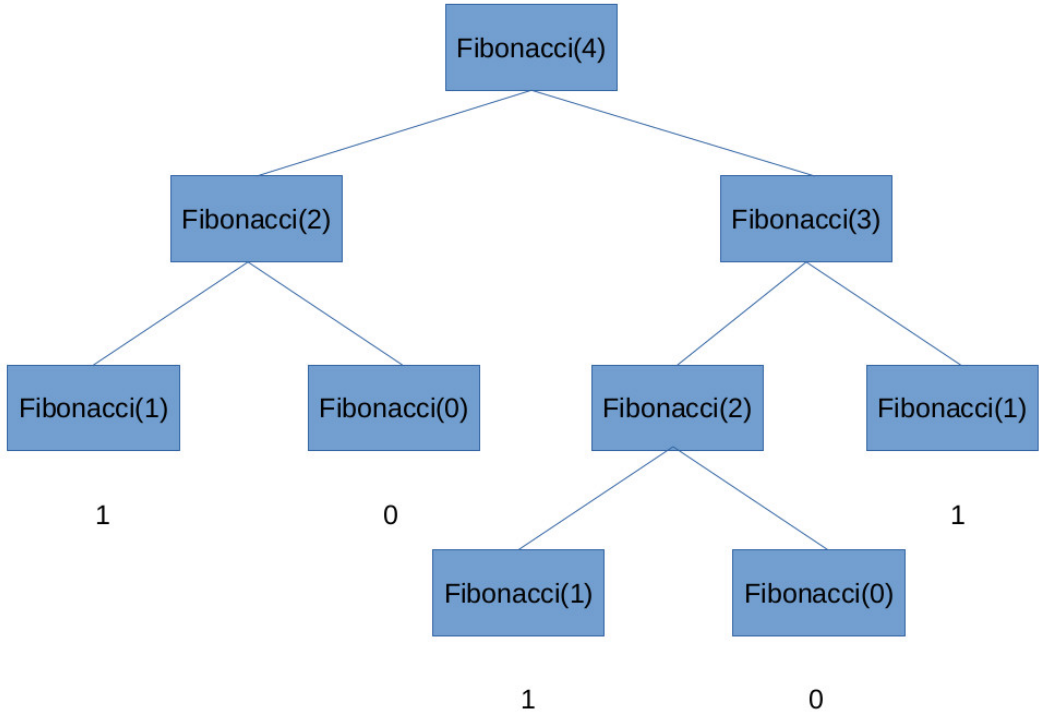
Na matemática, a Sequência de Fibonacci, é uma sequência de números inteiros, começando por 0, na qual, cada termo subsequente corresponde à soma dos dois anteriores. Os números de Fibonacci são, portanto, os números que compõem a seguinte sequência:

0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...

Após dominar o uso de funções recursivas envolvendo fibonacci, Charlinho ficou com uma pergunta na cabeça: "Quantas vezes cada chamada para a função fibonacci é realizada para cada número menor que n?" Para ajudar a responder a essa pergunta, ele resolveu desenhar uma estratura que parece uma árvore de ponta cabeça com as quantidades de chamadas, e ela ficou da seguinte forma:

Aviso!

?



Perceba que na árvore, Charlinho coloca no topo o valor que deseja calcular (Fibonacci de 4). Essa chamada gera outras duas chamadas para Fibonnaci de 2 e 3. A chamada de Fibonacci de 2 gera as chamadas de Fibonacci de 1 e 0 enquanto a chamada de Fibonacci de 3 geram as chamadas de Fibonacci de 2 e 1. Por fim, a chamada de Fibonacci de 2 geram as chamadas de Fibonacci de 1 e 0.

Para não precisar sempre desenhar essa árvore, ele pediu a sua ajuda para implementar um programa que conte quantas vezes cada chamada da função **`fibonacci`** é realizada, para cada valor de entrada diferente da função recursiva. Observe que, ao elaborar seu programa, você deve definir uma função **fibonacci** que recebe como parâmetro um número inteiro n fornecido da entrada padrão, e considerá-la na resolução do problema.

Entrada

A entrada consiste em um único inteiro $1 \leq n \leq 30$ que indica a quantidade de termos da sequência de Fibonacci.

Aviso!

Saída

A saída deve conter uma linha com um inteiro indicando o n-ésimo termo da sequência de fibonacci, e em seguida outras $n + 1$ linhas com os quantitativos de vezes em que cada função é chamada para um determinado valor menor ou igual a n, conforme os exemplos.

Observações

- No terceiro exemplo de teste, o quarto termo da sequência de fibonacci é 3. Assim ``Termo: 3''. Para calcular o fibonacci de 4, calculam-se: 2 vezes o fibonacci de 0, 3 vezes o fibonacci de 1, 2 vezes o fibonacci de 2, 1 vez o fibonacci de 3 e 1 vez o fibonacci de 4.

For example:

Input	Result
0	Termo: 0 Quantidades: fibonacci(0) - 1
1	Termo: 1 Quantidades: fibonacci(0) - 0 fibonacci(1) - 1
4	Termo: 3 Quantidades: fibonacci(0) - 2 fibonacci(1) - 3 fibonacci(2) - 2 fibonacci(3) - 1 fibonacci(4) - 1

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
posicao_termo = int(input())

chamadas_funcao = [0]*(posicao_termo+1)

def fibonacci(n):
    global chamadas_funcao
    chamadas_funcao[n] += 1
```

Aviso!

```
  if n == 0:
    return 0
  elif n == 1:
    return 1
  else:
    return fibonacci(n-1) + fibonacci(n-2)
```

	Input	Expected	Got	
✔	0	Termo: 0 Quantidades: fibonacci(0) - 1	Termo: 0 Quantidades: fibonacci(0) - 1	✔
✔	1	Termo: 1 Quantidades: fibonacci(0) - 0 fibonacci(1) - 1	Termo: 1 Quantidades: fibonacci(0) - 0 fibonacci(1) - 1	✔
✔	4	Termo: 3 Quantidades: fibonacci(0) - 2 fibonacci(1) - 3 fibonacci(2) - 2 fibonacci(3) - 1 fibonacci(4) - 1	Termo: 3 Quantidades: fibonacci(0) - 2 fibonacci(1) - 3 fibonacci(2) - 2 fibonacci(3) - 1 fibonacci(4) - 1	✔

Aviso!

?

	Input	Expected	Got	
✔	12	Termo: 144 Quantidades: fibonacci(0) - 89 fibonacci(1) - 144 fibonacci(2) - 89 fibonacci(3) - 55 fibonacci(4) - 34 fibonacci(5) - 21 fibonacci(6) - 13 fibonacci(7) - 8 fibonacci(8) - 5 fibonacci(9) - 3 fibonacci(10) - 2 fibonacci(11) - 1 fibonacci(12) - 1	Termo: 144 Quantidades: fibonacci(0) - 89 fibonacci(1) - 144 fibonacci(2) - 89 fibonacci(3) - 55 fibonacci(4) - 34 fibonacci(5) - 21 fibonacci(6) - 13 fibonacci(7) - 8 fibonacci(8) - 5 fibonacci(9) - 3 fibonacci(10) - 2 fibonacci(11) - 1 fibonacci(12) - 1	✔
✔	7	Termo: 13 Quantidades: fibonacci(0) - 8 fibonacci(1) - 13 fibonacci(2) - 8 fibonacci(3) - 5 fibonacci(4) - 3 fibonacci(5) - 2 fibonacci(6) - 1 fibonacci(7) - 1	Termo: 13 Quantidades: fibonacci(0) - 8 fibonacci(1) - 13 fibonacci(2) - 8 fibonacci(3) - 5 fibonacci(4) - 3 fibonacci(5) - 2 fibonacci(6) - 1 fibonacci(7) - 1	✔

Aviso!

?

	Input	Expected	Got	
✔	16	Termo: 987 Quantidades: fibonacci(0) - 610 fibonacci(1) - 987 fibonacci(2) - 610 fibonacci(3) - 377 fibonacci(4) - 233 fibonacci(5) - 144 fibonacci(6) - 89 fibonacci(7) - 55 fibonacci(8) - 34 fibonacci(9) - 21 fibonacci(10) - 13 fibonacci(11) - 8 fibonacci(12) - 5 fibonacci(13) - 3 fibonacci(14) - 2 fibonacci(15) - 1 fibonacci(16) - 1	Termo: 987 Quantidades: fibonacci(0) - 610 fibonacci(1) - 987 fibonacci(2) - 610 fibonacci(3) - 377 fibonacci(4) - 233 fibonacci(5) - 144 fibonacci(6) - 89 fibonacci(7) - 55 fibonacci(8) - 34 fibonacci(9) - 21 fibonacci(10) - 13 fibonacci(11) - 8 fibonacci(12) - 5 fibonacci(13) - 3 fibonacci(14) - 2 fibonacci(15) - 1 fibonacci(16) - 1	✔

Passou em todos os teste! ✔

Essa questão pode ser respondida criando a função fibonacci para fazer a chamada a uma função recursiva que efetivamente ira calcular o fibonacci do número e mostrar a resposta na tela. A função recursiva pode atualizar um contador em formato de lista onde cada posição corresponde ao fibonacci do número de mesmo valor da posição. Sempre que fibonacci de i for chamado o valor da posição i na lista é atualizado de 1 para contabilizar mais uma chamada a fibonacci de i.

No corpo principal do seu programa, basta chamar fibonacci(n) e imprimir o que foi solicitado.

Question author's solution (Python3):

```
1  quantidade = [0] * 31
2
3  def fibonacci(n):
4      quantidade[n] += 1;
5      if n == 0:
6          return 0
7      elif n == 1:
```

Aviso!

```
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input())

print(f'Termo: {fibonacci(n)}\nQuantidades:')
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão 8

Correto

Atingiu 1,00 de 1,00

42

A resposta de tudo. Mas qual a pergunta? Bom nesse caso a pergunta é se a soma de algum par de números em um vetor resulta em 42... e a resposta é sim ou não.

Entrada

A entrada consiste em 2 linhas. A primeira contém um inteiro n ($1 \leq 1000$)

A segunda linha contém n inteiros separados por espaço. Esses números não estão ordenados.

Saída

Imprima "**sim**" (sem aspas duplas) caso exista algum par de números cuja soma é 42. Imprima "**nao**" (sem aspas duplas) caso contrário.

Observações

- No primeiro caso de teste, temos os números 20 e 22, cuja soma é 42, então imprimimos **sim**.
- No segundo caso de teste, não existe nenhum par de números cuja soma é 42, então o resultado é **nao**.

For example:

Input	Result
10 22 10 42 5 20 12 13 15 94 100	sim
5 1 2 2 42 2	nao
18 61 12 25 8 88 25 90 69 90 13 62 7 84 40 54 27 2 50	sim

Aviso!

penalty regime: 0, 0, 10, 20, ... %)

?

```
1
2  qtd = int(input())
3  numeros = list(map(int, input().strip().split(' ')))
4
5  numeros.sort()
6
7
8  def verificaSoma(numeros):
9    for i in range(len(numeros)):
10     for j in range(i, len(numeros)):
11       if numeros[i]+numeros[j] == 42:
12         return "sim"
13   return "nao"
14
```

	Input	Expected	Got	
✔	10 22 10 42 5 20 12 13 15 94 100	sim	sim	✔
✔	5 1 2 2 42 2	nao	nao	✔
✔	18 61 12 25 8 88 25 90 69 90 13 62 7 84 40 54 27 2 50	sim	sim	✔
✔	32 -27 27 -68 -11 -37 -38 28 -81 37 -70 42 -66 43 28 24 33 49 32 32 27 31 -71 -7 -81 45 85 0 -43 -85 47 63 19	sim	sim	✔

Aviso!

	Input	Expected	Got	
✔	562 56 0 61 76 10 2 93 36 95 83 63 48 10 68 0 11 76 85 67 16 10 53 100 3 76 31 55 86 97 29 29 54 9 82 35 86 81 81 25 9 86 5 80 73 91 24 46 70 16 38 71 86 94 22 35 6 77 29 62 1 51 90 59 65 32 49 44 91 39 10 65 25 27 33 55 70 23 13 28 14 60 35 90 59 26 59 39 21 72 44 23 24 4 71 0 17 55 43 47 11 78 32 97 78 11 79 19 19 65 70 88 68 51 72 14 82 59 44 78 84 9 11 69 36 6 6 28 22 35 40 23 35 36 24 68 8 24 96 44 73 55 24 98 99 58 7 7 96 67 67 29 100 61 28 63 31 88 14 32 87 44 1 76 57 94 40 67 86 51 22 30 97 77 84 52 39 21 62 73 61 14 5 27 95 44 32 3 78 72 35 82 74 90 20 69 81 68 97 31 14 48 71 61 61 88 41 85 90 97 54 60 62 73 56 44 67 93 61 11 19 29 93 25 24 11 54 78 35 41 77 3 46 79 86 89 70 73 30 49 16 74 44 71 48 92 68 48 35 82 12 35 24 93 78 81 26 16 19 37 25 87 47 65 81 13 64 43 47 27 7 7 100 46 40 46 71 27 92 26 12 88 68 40 54 2 46 30 48 89 88 98 61 71 90 39 52 41 84 51 75 49 63 71 16 88 15 23 11 79 67 27 66 31 42 62 22 4 61 29 36 11 94 97 72 18 42 69 20 6 70 64 94 36 73 68 82 49 64 68 80 15 77 7 6 3 6 98 54 65 91 79 35 93 40 32 28 91 71 78 41 81 93 65 9 74 61 56 30 30 23 68 24 12 65 99 34 79 42 66 36 72 16 44 78 65 48 85 62 60 86 5 31 54 8 89 83 97 67 82 3 95 93 54 75 25 20 10 77 2 97 35 21 71 13 87 22 14 57 55 8 97 76 52 40 77 11 29 65 72 79 26 73 40 73 57 13 58 45 61 21 71 30 22 87 63 20 89 6 46 36 93 9 6 85 76 70 3 82 32 14 23 76 77 47 98 24 54 35 49 91 39 0 51 3 12 23 90 15 4 14 74 2 9 45 55 18 83 29 88 0 59 8 29 57 37 12 67 68 5 40 66 1 23 49 48 31 47 75 78 83 7 46 3 83 74 54 45 97 27 49 19 2 80 8 58 71 91 81 47 32 17 13 8 25 61 31 94 99 45 29 29 14 56 68 33 2 62 27 78 18 57 96 79 83 23 40 18 33 53 27 3 89	sim	sim	✔
✔	41 73 -21 49 -61 -46 0 -69 84 19 60 59 -85 -30 64 -71 -100 -32 80 -34 60 -18 -19 14 98 100 30 9 21 -24 -8 1 -95 -62 12 14 -73 18 39 -49 55 -29	sim	sim	✔

Passou em todos os teste! ✔

Existem 3 formas de resolver esse problema.

A primeira, e mais óbvia, é tentar todas as somas possíveis entre 2 números do vetor, e se alguma der 42, respondemos **sim**. Se testamos todas as somas e nenhuma deu 42, a resposta é **nao**. Essa forma é simples de ser implementada, mas é pouco eficiente.

Uma segunda alternativa é **ordenar** o vetor. Feito isso, podemos utilizar uma técnica chamada two pointers para descobrir se é possível somar 2 números até 42 da seguinte maneira:

1. Ordenar o vetor
2. Inicializar um ponteiro $L = 0$ e $R = n - 1$ (primeiro e último elementos do vetor

Aviso!

?

3. Enquanto o ponteiro L for menor do que R, analisamos a soma atual. Se ela for 42, retornamos **sim**. Se ela for **menor** do que 42, então precisamos de números maiores na soma, logo **incrementamos** L uma posição. Se a soma for **maior** que 42, precisamos de números menores, logo **decrementamos** R uma posição.

4. Se $L == R$ e não achamos uma soma que dá 42, podemos responder **nao**.

A terceira maneira de responder a questão envolve usar um mapa para armazenar quais números foram vistos. Quando você aprender sobre dicionários em Python pode tentar resolver essa questão novamente usando eles :D

Question author's solution (Python3):

```python
def solve(arr):
    arr.sort()
    L = 0
    R = len(arr) - 1
    while L < R:
        if arr[L] + arr[R] == 42:
            return "sim"
        elif arr[L] + arr[R] < 42:
            L += 1
        else:
            R -= 1
    return "nao"

n = int(input())
nums = input().split(' ')
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão **9**

Correto

Atingiu 1,00 de 1,00

Escolha Premiada

Você foi convidado para a chance ÚNica de Brindes do programa do Saustão. Esse jogo funciona da seguinte maneira, você vai receber uma lista de possíveis prêmios que poderá ganhar e cada um terá o seu respectivo valor. Depois de ver todos os prêmios, Sausto Filva irá sortear um valor para você. Se você conseguir escolher um número de brindes cujo a soma do valor desses brindes seja igual ao valor sorteado, você ganha todos os brindes, senão perde tudo. Como você já sabe como o jogo funciona e é um exímio programador, você resolveu implementar um programa para dizer se é possível escolher um conjunto de brindes que o faça ganhar o prêmio ou não.

Entrada

A entrada contém 2 linhas. A primeira linha contém n inteiros v onde cada v_i indica o valor de um dos prêmios da lista ($0 \leq v_i \leq 100000$ e $1 \leq n \leq 20$). A segunda linha contém um inteiro que indica o número s sorteado por Sausto ($1 \leq s \leq 2000000$).

Saída

A saída deve conter a frase 'E possivel ganhar.', se for possível ganhar os prêmios e 'Impossivel ganhar.' se não for possível ganhar os prêmios.

Notas

- No primeiro exemplo de teste, os brindes 2 e 4 possuem valores iguais a 2 e 8 que, somados, são iguais ao número sorteado;
- No segundo exemplo de teste, nenhuma combinação possível dos valores dos brindes disponíveis resulta em uma soma igual ao número sorteado por Saustão.

For example:

Input	Result
Aviso!	`E possivel ganhar.`

Input	Result
1 2 3 8 7	Impossivel ganhar.
2 2 2 2 5 2 2 15	E possivel ganhar.

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
numeros = list(map(int, input().strip().split(' ')))
valor = int(input())
import itertools


def ePossivelQtdValores(tamanho):
  dados = itertools.combinations(numeros, tamanho)
  sublistas = list(dados)
  for lista in sublistas:
    soma = 0
    for item in lista:
      soma += item
    if soma == valor:
      return True
  return False
```

	Input	Expected	Got	
✔	1 2 3 8 10	E possivel ganhar.	E possivel ganhar.	✔
✔	1 2 3 8 7	Impossivel ganhar.	Impossivel ganhar.	✔

Aviso!

	Input	Expected	Got	
✔	2 2 2 2 5 2 2 15	E possivel ganhar.	E possivel ganhar.	✔
✔	8 0 9 8 5 17	E possivel ganhar.	E possivel ganhar.	✔
✔	4 5 7 8 7 10	Impossivel ganhar.	Impossivel ganhar.	✔
✔	2 9 9 2 10 2 18 14 27	E possivel ganhar.	E possivel ganhar.	✔

Passou em todos os teste! ✔

Para resolver a questão, deve-se realizar todas as combinações possíveis de brindes e verificar se o valor somado deles resulta no valor do número sorteado s. A ideia é a seguinte: suponha um conjunto B, em que vamos adicionar os brindes que resultam no valor s. Vamos testar incluir um brinde i em B, de maneira a soma dos valores dos brindes em B não ultrapassem o número sorteado s. Tal abordagem é uma versão simples do Problema da Mochila, que pode ser resolvida por meio de uma abordagem recursiva. Assim, ao analisarmos um brinde i:

- se o valor do brinde i, juntamente com o valor total dos brindes em B, não ultrapassam o valor s, podemos incluir ou não o brinde i em B.
- agora, se a soma dos valores do brinde (\i\) com os valores dos brindes que já estão em B ultrapassam o valor s, não podemos incluir o brinde i em B.

Pode-se então elaborar uma função recursiva que, em cada chamada, testa se esse brinde i pode ser colocado em B, sem que a soma de todos os brindes em (\B\) (incluindo o brinde i) exceda s. Em caso afirmativo, podemos realizar duas novas chamadas nessa função: para o caso em que o brinde i é colocado em B; e para o caso em que o brinde i não é colocado em B. O critério de parada das recursões se baseia na situação em que a soma dos brindes em B resulta no número (\s\).

Question author's solution (Python3):

```
# Como valores possui escopo global, ele está acessível dentro da função verifica
def verifica(n, premio):

    if premio == 0:
```

Aviso!

```
        return True
    elif (n < 0) or (premio < 0):
        return False
    else:
        t1 = verifica(n - 1, premio - valores[n])
        t2 = verifica(n - 1, premio)

    return t1 or t2

valores = [int(s) for s in input().split()]
premio = int(input())
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!

?

Questão 10

Correto

Atingiu 1,00 de 1,00

Números binários

Durante suas pesquisas, a profa. Vanessa propôs o padrão *"apc-2021"* para número binários, visando representar números decimais positivos e menores do que 1. Na *"apc-2021"*, o número é representado por uma sequência de 0's e 1's de comprimento arbitrário. Lendo a representação da esquerda para a direita, o primeiro dígito binário representa o valor 2^{-1}, o segundo representa 2^{-2}, o terceiro 2^{-3}, e assim por diante. A representação utiliza sempre o menor número de dígitos possível (ou seja, desta forma o dígito mais à direita é sempre 1).

Por exemplo, a sequência de dígitos binários 01_2 representa o seguinte valor:

$$0*2^{-1} + 1*2^{-2} = 0.25$$

Já a sequência de dígitos binários 101011_2 representa o seguinte valor:

$$1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 0*2^{-4} + 1*2^{-5} + 1*2^{-6} = 0.671875$$

Sua tarefa é, dados dois números inteiros X e Y, representados no padrão "*apc-2021*", determinar a representação da soma $X + Y$, também no padrão *"apc-2021"*.

Entrada

A primeira linha contém os inteiros M e N ($1 \leq M, N \leq 10^3$), representando respectivamente o número de dígitos binários de X e de Y ($0 < X, Y < 1$). A segunda linha contém M números X_i ($X_i \in \{0, 1\}$), representando X no padrão *"apc-2021"*. A terceira linha contém N números Y_j ($Y_j \in \{0, 1\}$), representando Y no padrão *"apc-2021"*.

Saída

Aviso! ama deve produzir uma única linha, contendo a representação do valor $X + Y$ no padrão *"apc-2021"*, em que $X + Y < 1$.

?

Notas

- No primeiro exemplo de teste, a soma entre 100_2 e 10011_2 resulta em 10111_2.

For example:

Input	Result
3 5 0 0 1 1 1 0 0 1	1 1 1 0 1
6 6 1 0 1 0 0 1 1 1 0 0 1 1	0 1 1 1
5 4 1 0 1 1 1 0 0 0 1	1 1 0 0 1

Answer: (penalty regime: 0, 0, 10, 20, ... %)

```
tam1, tam2 = map(int, input().split(' '))
tamanhoMax = max(tam1, tam2)
num1, num2= [0]*tamanhoMax, [0]*tamanhoMax

num1[:tam1] = map(int, input().split(' '))
num2[:tam2] = map(int, input().split(' '))


def soma(num1, num2):
  resultado = [0]*tamanhoMax
  for x in reversed(range(tamanhoMax)):
    soma = num1[x] + num2[x] + resultado[x]
    if soma > 1:
      if x-1 >= 0:
```

Aviso!

?

	Input	Expected	Got	
✔	3 5 0 0 1 1 1 0 0 1	1 1 1 0 1	1 1 1 0 1	✔
✔	6 6 1 0 1 0 0 1 1 1 0 0 1 1	0 1 1 1	0 1 1 1	✔
✔	5 4 1 0 1 1 1 0 0 0 1	1 1 0 0 1	1 1 0 0 1	✔
✔	6 2 1 1 0 1 1 1 1 1	1 0 0 1 1 1	1 0 0 1 1 1	✔
✔	1 1 0 0	0	0	✔
✔	9 4 1 0 1 0 1 1 0 0 1 1 1 0 1	0 1 1 1 1 1 0 0 1	0 1 1 1 1 1 0 0 1	✔

Passou em todos os teste! ✔

Sejam os números binários $X = X_1, X_2, \ldots, X_m$ e $Y = Y_1, Y_2, \ldots, Y_n$ fornecidos na entrada. Primeiramente, deve-se completar com zeros mais à direita, o número binário entre X e Y de menor comprimento, tomando-se o cuidado de que na entrada, o bit mais à esquerda X_1 ou Y_1 do número binário é o mais significativo. Para resolver o problema, deve-se somar bit a bit, isto é, $X_i + Y_i$, para $i = max(m, n), \ldots, 0$, tratando-se o caso em que $1 + 1 = 10_2$, e que o bit 1 fica é contabilizado na soma da próxima posição mais significativa, isto é, para os bits na posição $i - 1$. Ao imprimir a resposta, desconsidere a sequência de zeros que aparecem na parte mais significativa antes do bit 1 menos significativo.

Question author's solution (Python3):

Aviso!

```
m,n = (int(x) for x in input().split())
```

```
2
3  bin1 = [int(x) for x in input().split()]
4  bin2 = [int(x) for x in input().split()]
5
6  if m > n:
7      bin2=bin2+([0]*(m-n))
8  else:
9      bin1=bin1+([0]*(n-m))
10
11 maior = max(m,n)
12
13 bin3=[0]*maior
14
15 acc = 0
```

Correto

Notas para este envio: 1,00/1,00.

Aviso!