Iniciado em	domingo, 31 out 2021, 11:43
Estado	Finalizada
Concluída em	domingo, 31 out 2021, 20:40
Tempo empregado	8 horas 56 minutos
Avaliar	**10,00** de um máximo de 10,00(**100**%)

Aviso!

?

Questão 1

Correto

Atingiu 1,00 de 1,00

World Hello?

Crie uma função **aprendendoArquivos** que recebe uma string s representando o nome do arquivo que você deverá usar. Abra o arquivo e verifique: se a primeira linha for um único inteiro, imprima "42 eh a resposta para tudo!", caso contrário imprima "Ola t!", sendo t a primeira linha do arquivo.

Entrada

- A entrada consiste no parâmetro da função **aprendendoArquivos**, que é uma única string s, $1 \leq |s| \leq 20$.
- É garantido que a string termina em ".txt".

Saída

Sua função deve imprimir de acordo com o especificado.

For example:

Test	Result
`aprendendoArquivos("a.txt")`	`42 eh a resposta para tudo!`
`aprendendoArquivos("b.txt")`	`Ola Professor Lindosvau!`
`aprendendoArquivos("c.txt")`	`Ola 4 muy lindo!`

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

```
def aprendendoArquivos(fileName):
  with open(fileName, 'r', encoding='utf-8') as file:
    firstLine = file.readline()
    if firstLine[-1] == '\n':
      firstLine = firstLine[:-1]
```

Aviso!

```
    try:
      numero = int(firstLine)
      print('42 eh a resposta para tudo!')
    except ValueError:
      print(f'Ola {firstLine}!')
```

	Test	Expected	Got	
✔	aprendendoArquivos("a.txt")	42 eh a resposta para tudo!	42 eh a resposta para tudo!	✔
✔	aprendendoArquivos("b.txt")	Ola Professor Lindosvau!	Ola Professor Lindosvau!	✔
✔	aprendendoArquivos("c.txt")	Ola 4 muy lindo!	Ola 4 muy lindo!	✔
✔	aprendendoArquivos("d.txt")	Ola sakdhjlash iwhefiwhe fajhf 42!	Ola sakdhjlash iwhefiwhe fajhf 42!	✔
✔	aprendendoArquivos("e.txt")	42 eh a resposta para tudo!	42 eh a resposta para tudo!	✔
✔	aprendendoArquivos("f.txt")	Ola sdkhj 3ijhfkjwhe 2ih kjqwhe kjfh!	Ola sdkhj 3ijhfkjwhe 2ih kjqwhe kjfh!	✔

Passou em todos os teste! ✔

Para resolver o problema, deve-se abrir o arquivo e verificar se a primeira linha do arquivo é um número ou não.

Question author's solution (Python3):

```
def aprendendoArquivos(s):
    with open(s, 'r') as f:
        linha = f.readline()
```

Aviso!

```
4       # print(linha)
5       try:
6           int(linha)
7           print('42 eh a resposta para tudo!')
8       except:
9           #linha = linha[:-1]
10          if linha[-1] == '\n':
```

Correto

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

Aviso!

?

Questão **2**

Correto

Atingiu 1,00 de 1,00

Checagem

A entrada consiste em duas strings s, t representando o nome de 2 arquivos. Verifique qual arquivo possui exatamente um caractere e imprima uma linha com a frase "O arquivo x esta quase vazio, mas o y nao esta!" em que x é o nome do arquivo quase vazio e y é o nome do arquivo não vazio. Após essa linha imprima o conteúdo de y.

Entrada

- A entrada contém duas strings s, t, $1 \leq |s|, |t| \leq 20$.
- Em relação ao nome dos arquivos, é garantido que s e t terminam em ".txt".
- É garantido que exatamente um arquivo está quase vazio e o outro não está.

Saída

Imprima de acordo com o enunciado.

For example:

Input	Result
a.txt t.txt	O arquivo t.txt esta quase vazio, mas o a.txt nao esta! esse conteudo esta no arquivo a.txt (?)

Aviso!

Input	Result
b.txt s.txt	O arquivo b.txt esta quase vazio, mas o s.txt nao esta! lkdjhflkakjd asodf ja[fw4 g5asg6a s ga6s d6asd fa6 s5d4f6as54g65r ga6 r6 h a6 hr h6arh4a6d4h6a8h8ah46 aha6f h 6adf4 6a4 6f4h 6adfh4 6a5df1h32 fd16df8h ad2fh 6e5h 6a5d4h 6ad45fh6a d4fh6 a4h54daf6 h54a6fg 3adf54g 6a fg6af5d h6a5fd h5 adf65h ad5f h 3ad5f h a35df h6a d5f h a6
c.txt r.txt	O arquivo r.txt esta quase vazio, mas o c.txt nao esta! bom dia caros amigos como estao?

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

```
f1, f2 = input().strip().split(' ')

with open(f1, 'r', encoding='utf-8') as file1, open(f2, 'r', encoding='utf-8') as file2:
  content1, content2 = file1.read(), file2.read()
  if len(content1) == 1:
    print(f'O arquivo {f1} esta quase vazio, mas o {f2} nao esta!\n{content2}')
  elif len(content2) == 1:
    print(f'O arquivo {f2} esta quase vazio, mas o {f1} nao esta!\n{content1}')


```

Aviso!

	Input	Expected	Got	
✔	a.txt t.txt	O arquivo t.txt esta quase vazio, mas o a.txt nao esta! esse conteudo esta no arquivo a.txt (?)	O arquivo t.txt esta quase vazio, mas o a.txt nao esta! esse conteudo esta no arquivo a.txt (?)	✔
✔	b.txt s.txt	O arquivo b.txt esta quase vazio, mas o s.txt nao esta! lkdjhflkakjd asodf ja[fw4 g5asg6a s ga6s d6asd fa6 s5d4f6as54g65r ga6 r6 h a6 hr h6arh4a6d4h6a8h8ah46 aha6f h 6adf4 6a4 6f4h 6adfh4 6a5df1h32 fd16df8h ad2fh 6e5h 6a5d4h 6ad45fh6a d4fh6 a4h54daf6 h54a6fg 3adf54g 6a fg6af5d h6a5fd h5 adf65h ad5f h 3ad5f h a35df h6a d5f h a6	O arquivo b.txt esta quase vazio, mas o s.txt nao esta! lkdjhflkakjd asodf ja[fw4 g5asg6a s ga6s d6asd fa6 s5d4f6as54g65r ga6 r6 h a6 hr h6arh4a6d4h6a8h8ah46 aha6f h 6adf4 6a4 6f4h 6adfh4 6a5df1h32 fd16df8h ad2fh 6e5h 6a5d4h 6ad45fh6a d4fh6 a4h54daf6 h54a6fg 3adf54g 6a fg6af5d h6a5fd h5 adf65h ad5f h 3ad5f h a35df h6a d5f h a6	✔
✔	c.txt r.txt	O arquivo r.txt esta quase vazio, mas o c.txt nao esta! bom dia caros amigos como estao?	O arquivo r.txt esta quase vazio, mas o c.txt nao esta! bom dia caros amigos como estao?	✔

Aviso!

?

	Input	Expected	Got	
✔	d.txt q.txt	O arquivo d.txt esta quase vazio, mas o q.txt nao esta! akjdfhaslkdhj fasd a5s7df 5sa4d g 5as4 g a54f g5a4f ga5f4 gag 6as5d4g6sa gas g a3s5 g sa4g 6as g 6as g6 as54g a6s5f4g6as4f6g a4f64h d654h 6ad5g4h 6daha 6d fh 6adf h6 ad5f h6da5 4h a6df 5ah6d f6h5adf4ha6d f4h65d46h ad468fh 6d5f4h3a d5f45h a	O arquivo d.txt esta quase vazio, mas o q.txt nao esta! akjdfhaslkdhj fasd a5s7df 5sa4d g 5as4 g a54f g5a4f ga5f4 gag 6as5d4g6sa gas g a3s5 g sa4g 6as g 6as g6 as54g a6s5f4g6as4f6g a4f64h d654h 6ad5g4h 6daha 6d fh 6adf h6 ad5f h6da5 4h a6df 5ah6d f6h5adf4ha6d f4h65d46h ad468fh 6d5f4h3a d5f45h a	✔
✔	e.txt p.txt	O arquivo p.txt esta quase vazio, mas o e.txt nao esta! akjdhsfk dhfkashj dkajhs dfkjahsd flasdhjgiwhihaslkfhruiyhvksadjv nossa mas como é que foi parar nesse assunto e como foi parar nesse assunto mas tudo bem osh o que foi sim foi por que voce eh inteligente genial gentil esbelta	O arquivo p.txt esta quase vazio, mas o e.txt nao esta! akjdhsfk dhfkashj dkajhs dfkjahsd flasdhjgiwhihaslkfhruiyhvksadjv nossa mas como é que foi parar nesse assunto e como foi parar nesse assunto mas tudo bem osh o que foi sim foi por que voce eh inteligente genial gentil esbelta	✔
✔	f.txt o.txt	O arquivo f.txt esta quase vazio, mas o o.txt nao esta! aksdhjlak fhakjdhf lakshjf alkefh waef ajdhfkasdhfkasdhjf askdhjf aksdjhfalkdhfl kh f akljhfalkjhsfl kah fihud fiahd faodf akjhfalksdhf lkAJHDF LAKHDF KAH sd 13d1f65a4df6as54df 6adf465asdf65 1a65dg1h1 65j1	O arquivo f.txt esta quase vazio, mas o o.txt nao esta! aksdhjlak fhakjdhf lakshjf alkefh waef ajdhfkasdhfkasdhjf askdhjf aksdjhfalkdhfl kh f akljhfalkjhsfl kah fihud fiahd faodf akjhfalksdhf lkAJHDF LAKHDF KAH sd 13d1f65a4df6as54df 6adf465asdf65 1a65dg1h1 65j1	✔

Aviso!

m todos os teste! ✔

?

Para resolver o problema, deve-se verificar qual arquivo está vazio e imprimi-lo.

Question author's solution (Python3):

```
import os
s, t = input().split()

s_size = os.path.getsize(s)
t_size = os.path.getsize(t)

print(f"O arquivo {s if t_size > 1 else t} esta quase vazio, mas o {s if s_size > 1 else t
```

Correto

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

Aviso!

?

Questão 3

Correto

Atingiu 1,00 de 1,00

Presença

Reprovar por falta é uma das coisas mais tristes que pode acontecer com um aluno. Infelizmente é uma prática que ainda acontece. O exercício hoje é descobrir quais alunos reprovaram por falta.

Lembre-se que um aluno é reprovado por falta se sua presença é menor que 75% do total de aulas que foram ministradas.

Entrada

A entrada do programa consiste de uma linha contendo o nome do arquivo CSV (valores separados por vírgula - *comma-separated values*) a ser lido. Esse arquivo é formatado como um CSV normal, em que cada linha é uma sequência de nomes separados por vírgula. Cada nome é um aluno da turma. Cada linha é a presença de uma aula que foi dada.

Saída

Imprima o nome de todos os alunos reprovados por falta. Todos os nomes devem estar na mesma linha separados por uma espaço, por exemplo $a_1\ a_2\ a_3$, onde a_i é o nome de um aluno.

Se nenhum aluno reprovar, imprima "Nenhum aluno reprovado por faltas".

Notas

O primeiro exemplo usa o conteúdo abaixo

```
Luiza,Murilo,Anitta,Aline,Laferte
Laferte,Aline,Luiza,Anitta
Laferte,Aline,Luiza,Anitta
Luiza,Aline,Anitta,Laferte
Aline,Anitta,Laferte,Murilo
```

Note que Murilo só esteve presente em 2 das 5 aulas. Isso representa 40% das aulas, que é menos que os 75% exigidos.

For example:

Input	Result
0.csv	Murilo
	uliana

Aviso!

?

Input	Result
2.csv	Xavier

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

```
filePath = input().strip()

with open(filePath, 'r', encoding='utf-8') as chamada:
  aulas = chamada.readlines()

presencas = []

for dia in aulas:
  presencas.extend(dia.replace('\n', '').split(','))

def getPresencas():
  naoRepetidos = []
  for nome in presencas:
    if nome not in naoRepetidos:
      naoRepetidos.append(nome)
```

	Input	Expected	Got	
✔	0.csv	Murilo	Murilo	✔
✔	1.csv	Juliana	Juliana	✔
✔	2.csv	Xavier	Xavier	✔
✔	3.csv	Fernanda Lucas Cinira Sarah	Fernanda Lucas Cinira Sarah	✔

Aviso!

	Input	Expected	Got	
✔	4.csv	Nenhum aluno reprovado por faltas	Nenhum aluno reprovado por faltas	✔
✔	5.csv	Nenhum aluno reprovado por faltas	Nenhum aluno reprovado por faltas	✔

Passou em todos os teste! ✔

Precisamos de um dicionário para manter a frequência de cada aluno. A chave é o nome do aluno, e o valor é a quantidade de vezes que vemos o nome dele ao longo do arquivo, já que o nome dele aparece no máximo uma vez por linha (= por aula).

A segunda informação importante é a quantidade de linhas no arquivo, que equivale a quantidade de aulas dadas.

Finalmente, basta ver se a frequência dos alunos é menor que 75% (e.g. $frequencia[aluno]/total\_aulas < 0.75$), para saber quem está reprovado.

PS.: É necessário fazer o stripping das linhas para evitar a quebra de linha no nome do último aluno

Question author's solution (Python3):

```
csv = input()

def reprovados(lines):
    days_of_class = len(lines)
    attendance = {}
    for l in lines:
        names = l.strip().split(',')
        for n in names:
            attendance[n] = attendance.get(n, 0) + 1
    return [k for k, v in attendance.items() if v / days_of_class < 0.75]

with open(csv, 'r') as file:
    lines = file.readlines()
    r = reprovados(lines)
    if len(r) == 0:
```

Aviso!

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

Aviso!

?

Questão **4**

Correto

Atingiu 1,00 de 1,00

Classificador de Filmes

Crie um programa que lê o arquivo $filmes.txt$ e retorna os filmes de um gênero e duração que o usuário pode ter interesse. O programa recebe como entradas uma string $genero$, que é utilizada para pesquisar o gênero de filme desejado, e um inteiro $duracao$, que é a duração máxima, em minutos, que o filme procurado deve ter.
Cada linha do arquivo $filmes.txt$ é composta pelo nome do filme, seu gênero e sua duração (em minutos), separados pelo caractere */*.
Por fim, liste todos os filmes que tenham o gênero procurado e duração menor ou igual ao desejado.
Caso não existam filmes que se apliquem na entrada dada, imprima: "Não foram encontrados filmes assim".

Entrada
A entrada do programa consiste em uma string e um inteiro.

Saída
A saída do programa consiste nos filmes que sigam as condições da entrada.

Observação
Não diferencie entre letras maiúsculas e minúsculas.
For example:

Input	Result
ação 300	Batman: O Cavaleiro das Trevas A Origem Matrix Os Sete Samurais O Profissional Harakiri O Exterminador do futuro 2: O julgamento final Gladiador

Aviso!

?

Input	Result
drama 120	Whiplash: Em Busca da Perfeição Casablanca
drama 180	Um Sonho de Liberdade O Poderoso Chefão 12 Homens e uma Sentença Pulp Fiction: Tempo de Violência Clube da Luta Forrest Gump: O Contador de Histórias Os Bons Companheiros Um Estranho no Ninho Cidade de Deus A Felicidade Não se Compra O Resgate do Soldado Ryan O Pianista A Outra História Americana Whiplash: Em Busca da Perfeição Casablanca

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

```
1  with open('filmes.txt', 'r', encoding='utf-8') as filmes:
2    content = filmes.readlines()
3
4  #A Origem/ação/250/n
5
6  filmes = dict()
7  GENERO, DURACAO = 0, 1
8
9  for filme in content:
10   dados = filme.replace('\n', '').split('/')
11   filmes[dados[0]] = [dados[1], int(dados[2])]
12
13 generoDesejado, tempoMax = input(), int(input())
```

Aviso!

	Input	Expected	Got	
✔	ação 300	Batman: O Cavaleiro das Trevas A Origem Matrix Os Sete Samurais O Profissional Harakiri O Exterminador do futuro 2: O julgamento final Gladiador	Batman: O Cavaleiro das Trevas A Origem Matrix Os Sete Samurais O Profissional Harakiri O Exterminador do futuro 2: O julgamento final Gladiador	✔
✔	drama 120	Whiplash: Em Busca da Perfeição Casablanca	Whiplash: Em Busca da Perfeição Casablanca	✔
✔	drama 180	Um Sonho de Liberdade O Poderoso Chefão 12 Homens e uma Sentença Pulp Fiction: Tempo de Violência Clube da Luta Forrest Gump: O Contador de Histórias Os Bons Companheiros Um Estranho no Ninho Cidade de Deus A Felicidade Não se Compra O Resgate do Soldado Ryan O Pianista A Outra História Americana Whiplash: Em Busca da Perfeição Casablanca	Um Sonho de Liberdade O Poderoso Chefão 12 Homens e uma Sentença Pulp Fiction: Tempo de Violência Clube da Luta Forrest Gump: O Contador de Histórias Os Bons Companheiros Um Estranho no Ninho Cidade de Deus A Felicidade Não se Compra O Resgate do Soldado Ryan O Pianista A Outra História Americana Whiplash: Em Busca da Perfeição Casablanca	✔
✔	aventura 150	Não foram encontrados filmes assim	Não foram encontrados filmes assim	✔
✔	terror 120	Psicose Alien - O 8º Passageiro	Psicose Alien - O 8º Passageiro	✔
✔	drama 90	Não foram encontrados filmes assim	Não foram encontrados filmes assim	✔

Aviso!

?

	Input	Expected	Got	
✔	testando 999	Não foram encontrados filmes assim	Não foram encontrados filmes assim	✔

Passou em todos os teste! ✔

Question author's solution (Python3):

```
1  import os
2
3  arq = open('filmes.txt', 'r')
4  genero = input()
5  duracao = int(input())
6  cont = 0
7  for linha in arq:
8      palavras = linha.split("/")
9      if palavras[1].upper() == genero.upper() and int(palavras[2]) <= duracao:
10         print(palavras[0])
11         cont += 1
```

Correto

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

Aviso!

?

Questão 5

Correto

Atingiu 1,00 de 1,00

A princesa e o dragão

Uma história muito conhecida e bem clichê é a da princesa no quarto mais alto da torre mais alta guardada por um dragão. E geralmente tem um cavaleiro no meio, e essa não é diferente.

Temos uma imagem da torre onde estão a princesa (no quarto mais alto), um cavaleiro correndo por sua vida para chegar até a princesa, e um dragão com fome. Essa imagem está em um arquivo de texto como o exemplo abaixo

```
|¯|_|¯|_|¯|_|¯|_|¯|
|                P  |
|-----------------|
|                 |
|-----------------|
|        C        |
|-----------------|
|                 |
|-----------------|
|                 |
|-----------------|
|        D        |
###################
```

Veja que a princesa está no andar 6, o cavaleiro no andar 4 e o dragão no andar 1.

A torre **sempre** possui essa largura, e os andares são sempre desta forma, mas o tamanho da torre (quantidade de andares) pode ser diferente.

Além disso, o cavaleiro consegue subir 1 andar por vez, mas o dragão consegue subir 2 andares por vez. Como cavaleiros correndo atrás de princesas são uma ocasião cada vez mais rara o dragão ficou animado e disse alguma coisa no final da história, ou ele disse "Quero ver descerem u.u" - se o cavaleiro conseguiu chegar até a princesa - ou então "Mais um pro lanche!" - se ele conseguiu chegar ao cavaleiro antes.

O seu trabalho é dizer o que acontece nessa história.

Aviso!

Entrada

A entrada do programa consiste de uma linha contendo o nome do arquivo **txt** a ser lido. Ele é formatado conforme mostrado acima, com a imagem da torre.

É garantido que haverá uma princesa P, um cavaleiro C e um dragão D na torre. Eles estarão dentro de algum andar da torre (as linhas com espaços).

Saída

Você deve imprimir 4 linhas.

A primeira linha deve ser "Princesa no andar a_p", onde a_p é o andar da princesa.

A segunda linha deve ser "Cavaleiro no andar a_c", onde a_c é o andar do cavaleiro.

A terceira linha deve ser "Dragão no andar a_d", onde a_d é o andar do dragão.

A quarta linha deve ser a frase que o dragão diz no final da história. Se o dragão consegue chegar até o mesmo andar que o cavaliro, ou até um andar acima do cavaleiro **antes** que o cavaleiro chegue no andar da princesa, ele diz "**Mais um pro lanche!**". Se eles chegarem no último andar juntos então o cavaleiro vence, porque o dragão nunca entra no quarto da princesa. Se o cavaleiro chegar no quarto da princesa primeiro, ele diz "**Quero ver descerem u.u**"

Notas

- O primeiro exemplo utiliza o arquivo exemplo mostrado no enúnciado.

For example:

Input	Result

`0.txt`	`Princesa no andar 6`
	`Cavaleiro no andar 4`
	`Dragão no andar 1`
	`Quero ver descerem u.u`

`1.txt`	`Princesa no andar 7`
	`Cavaleiro no andar 5`
	`Dragão no andar 4`
	`Mais um pro lanche!`

Aviso!

Input	Result
2.txt	Princesa no andar 9 Cavaleiro no andar 6 Dragão no andar 2 Quero ver descerem u.u

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

```
filePath = input().strip()

PRINCESA, CAVALEIRO, DRAGAO = 'P', 'C', 'D'
with open(filePath, 'r', encoding='utf-8') as castelo:
  torre = castelo.readlines()

for i in range(len(torre)):
  if PRINCESA in torre[i]:
    locPrincesa = int((len(torre) - i)/2)
    print(f'Princesa no andar {locPrincesa}')
  if CAVALEIRO in torre[i]:
    locCavaleiro = int((len(torre) - i)/2)
    print(f'Cavaleiro no andar {locCavaleiro}')
  if DRAGAO in torre[i]:
```

	Input	Expected	Got	
✔	0.txt	Princesa no andar 6 Cavaleiro no andar 4 Dragão no andar 1 Quero ver descerem u.u	Princesa no andar 6 Cavaleiro no andar 4 Dragão no andar 1 Quero ver descerem u.u	✔

Aviso!

	Input	Expected	Got	
✔	1.txt	Princesa no andar 7 Cavaleiro no andar 5 Dragão no andar 4 Mais um pro lanche!	Princesa no andar 7 Cavaleiro no andar 5 Dragão no andar 4 Mais um pro lanche!	✔
✔	2.txt	Princesa no andar 9 Cavaleiro no andar 6 Dragão no andar 2 Quero ver descerem u.u	Princesa no andar 9 Cavaleiro no andar 6 Dragão no andar 2 Quero ver descerem u.u	✔
✔	3.txt	Princesa no andar 11 Cavaleiro no andar 10 Dragão no andar 3 Quero ver descerem u.u	Princesa no andar 11 Cavaleiro no andar 10 Dragão no andar 3 Quero ver descerem u.u	✔
✔	4.txt	Princesa no andar 11 Cavaleiro no andar 7 Dragão no andar 5 Mais um pro lanche!	Princesa no andar 11 Cavaleiro no andar 7 Dragão no andar 5 Mais um pro lanche!	✔
✔	5.txt	Princesa no andar 13 Cavaleiro no andar 8 Dragão no andar 5 Mais um pro lanche!	Princesa no andar 13 Cavaleiro no andar 8 Dragão no andar 5 Mais um pro lanche!	✔

Passou em todos os teste! ✔

Nesse problema, precisamos ler todas as linhas, porque a torre está "invertida" em relação ao índice das linhas do arquivo. Dado isso, a quantidade de andares é simplesmente $linhas_do_arquivo//2$. Esse é também o andar da princesa.

Para achar o andar do dragão e do cavaleiro vemos se alguma das linhas possui os caracteres 'C' ou 'D'. O andar de qualquer linha é $a_p - (i//2)$, onde a_p é o andar da princesa, e i o índice da linha em questão.

Finalmente, o dragão precisa de \((a_p - a_d) // 2\) "movimentos" para chegar no último andar. O cavaleiro precisa de $a_p - a_c$ "movimentos". Sempre que a quantidade de movimentos do dragão for maior que do cavaleiro, o cavaleiro consegue chegar até a princesa.

Aviso!

?

Question author's solution (Python3):

```python
txt = input()

def solve(lines):
    height = len(lines) // 2
    knight_floor = -1
    dragon_floor = -1
    for i, l in enumerate(lines):
        if l.find('C') != -1:
            knight_floor = (i // 2)
        if l.find('D') != -1:
            dragon_floor = (i // 2)
    if knight_floor == -1 or dragon_floor == -1:
        raise f'Missing player. C: {knight_floor} D: {dragon_floor}'
    return height, height - knight_floor, height - dragon_floor
```

Correto

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

Aviso!

?

Questão **6**

Correto

Atingiu 1,00 de 1,00

Conta palavras

Escreva uma função chamada **palavras_repetidas** que recebe o nome do arquivo e uma palavra. A função deve abrir esse arquivo e contar quantas vezes a palavra aparece no arquivo.

Entrada
Não há entrada de dados, os arquivos existem e a função é chamada automaticamente

Saída
A função deve imprimir o número de vezes que uma determinada palavra se repete, de acordo com os exemplos abaixo.

Notas
- Caracteres em caixa alta e em caixa baixa são tratados de modo diferente. O caso de teste 3 procura a palavra 'perfeita' no arquivo 'maquiavel.txt'. O documento tem somente a palavra 'perfeitamente' portanto o seu algoritmo deve considerar que encontrou a palavra.
- Submeta somente o que foi solicitado.

Particularidade do Tópico
Atenção, a criação de uma função com o nome determinado pelo enunciado é fundamental para a prática do aluno e o Moodle irá descontar pontos, por esse critério, caso a criação não tenha sido feita corretamente (sendo case-sensitive o nome da função).

For example:

Test	Result
`palavras_repetidas('estoria.txt', 'vez')`	`vez aparece no arquivo estoria.txt 1 vez(es).`
`_repetidas('gda.txt', 'Papa')`	`Papa aparece no arquivo gda.txt 0 vez(es).`

Aviso!

Test	Result
palavras_repetidas('maquiavel.txt', 'perfeita')	perfeita aparece no arquivo maquiavel.txt 1 vez(es).

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

```
def palavras_repetidas(filePath, palavra):
  qtdVezes = 0
  with open(filePath, 'r', encoding='utf-8') as arquivo:
    for line in arquivo:
      qtdVezes += line.count(palavra)
  return qtdVezes
def palavras_repetidas(filePath, palavra):
  qtdVezes = 0
  with open(filePath, 'r', encoding='utf-8') as arquivo:
    for line in arquivo:
      qtdVezes += line.count(palavra)
  print(f'{palavra} aparece no arquivo {filePath} {qtdVezes} vez(es).')
```

	Test	Expected	Got	
✔	palavras_repetidas('estoria.txt', 'vez')	vez aparece no arquivo estoria.txt 1 vez(es).	vez aparece no arquivo estoria.txt 1 vez(es).	✔
✔	palavras_repetidas('gda.txt', 'Papa')	Papa aparece no arquivo gda.txt 0 vez(es).	Papa aparece no arquivo gda.txt 0 vez(es).	✔
✔	palavras_repetidas('maquiavel.txt', 'perfeita')	perfeita aparece no arquivo maquiavel.txt 1 vez(es).	perfeita aparece no arquivo maquiavel.txt 1 vez(es).	✔
✔	palavras_repetidas('estoria.txt', 'ao')	Nao aparece no arquivo estoria.txt 1 vez(es).	Nao aparece no arquivo estoria.txt 1 vez(es).	✔

Aviso!

?

	Test	Expected	Got	
✔	`palavras_repetidas('maquiavel.txt', 'mal')`	`mal aparece no arquivo maquiavel.txt 3 vez(es).`	`mal aparece no arquivo maquiavel.txt 3 vez(es).`	✔
✔	`palavras_repetidas('gda.txt', 'vez')`	`vez aparece no arquivo gda.txt 1 vez(es).`	`vez aparece no arquivo gda.txt 1 vez(es).`	✔

Passou em todos os teste! ✔

Essa questão pode ser respondida iterando sobre o arquivo e lendo linha a linha. Em cada linha verificar se a palavra está presente. Caso positivo, atualize de uma unidade um contador de palavras (inicializado em zero.).

Outra forma, é iterar sobre o conteúdo do arquivo, iniciando em cada caractere subsequente e verificando se a primeira sequência de caracteres que aparece é a palavra procurada.

Question author's solution (Python3):

```
1 def palavras_repetidas(arquivo, palavra):
2     dados = open(arquivo, 'r').read()
3     tam = len(dados)
4     ans = 0
5     for i in range(tam):
6         if dados[i:].startswith(palavra):
7             ans += 1
```

Correto

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

Aviso!

?

Questão 7

Correto

Atingiu 1,00 de 1,00

Incrivelmente Fácil

Esse está sendo um semestre difícil. Para todos nós.

A aluna Tatá, que está sobrecarregada com o final do semestre, comentou com um dos tutores que "um exercício de arquivos incrivelmente fácil ia ser bom". E bom, só dessa vez, o pedido de Tatá será atendido!

Esse exercício requer que você tente abrir um arquivo txt. Sim, só isso. Abaixo será explicado como deverá ser feito isso:

Entrada
A entrada do programa consiste de uma frase, no seguinte modelo: *tente abrir o arquivo nomedoarquivo.txt,* onde *nomedoarquivo.txt* é um nome de arquivo arbitrário.

Saída
A saída do programa consiste em mostrar na tela a frase "**O arquivo existe**" caso seja pra abrir o arquivo, ou seja, é um arquivo que existe, ou "**Arquivo inexistente**" caso o arquivo não exista.

Notas
Para verificar a existência de um determinado arquivo, pode-se utilizar a biblioteca `os`, e a função `isfile`, da seguinte forma:

```
import os
os.path.isfile('nomeDoArquivo.txt')
```

A função `isfile` irá retornar verdadeiro, caso o nome dado seja de um arquivo.

For example:

Input	Result
`tenta abrir o arquivo vai_abrir.txt`	`O arquivo existe`
`ir o arquivo n_vai_abrir.txt`	`Arquivo inexistente`

Aviso!

Input	Result
tenta abrir o arquivo a.txt	O arquivo existe

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

```
import os

filePath = input().strip().split(' ')[-1]

if os.path.isfile(filePath):
  print('O arquivo existe')
else:
  print('Arquivo inexistente')
```

	Input	Expected	Got	
✔	tenta abrir o arquivo vai_abrir.txt	O arquivo existe	O arquivo existe	✔
✔	tenta abrir o arquivo n_vai_abrir.txt	Arquivo inexistente	Arquivo inexistente	✔
✔	tenta abrir o arquivo a.txt	O arquivo existe	O arquivo existe	✔
✔	tenta abrir o arquivo b.txt	O arquivo existe	O arquivo existe	✔

Aviso!

?

	Input	Expected	Got	
✔	tenta abrir o arquivo c.txt	Arquivo inexistente	Arquivo inexistente	✔
✔	tenta abrir o arquivo d.txt	O arquivo existe	O arquivo existe	✔

Passou em todos os teste! ✔

Essa questão pode ser solucionado capturando o input "*tente abrir o arquivo nomedoarquivo.txt*" e transformando em lista cujos elementos são cada palavra separada por espaço. Em seguida, basta verificar se o último elemento é um nome de um arquivo existente no diretório.

Question author's solution (Python3):

```
1  import os
2
3  frase = input().split()
4  if os.path.isfile(frase[-1]):
5      print('O arquivo existe')
6  else:
```

Correto

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

Aviso!

?

Questão 8

Correto

Atingiu 1,00 de 1,00

Diferentes

Pedro e !Pedro são irmãos gêmeos, mas diferente do que a maioria pensa, eles são completamente opostos! Um gosta de branco, outro de preto, um gosta de feijão, o outro de arroz, um gosta de correr, e o outro de nadar. Acho que deu pra entender o quão opostos eles são, certo? Porém, ambos gostam muito de programar, mas como era de se esperar, eles também discordam em muitos pontos. Para Pedro, um segmento de um vetor é $k - bonito$ se o maior elemento desse segmento é pelo menos duas vezes maior que o tamanho do segmento, mais formalmente, se n é o maior número no intervalo de tamanho k, então $2 \cdot k \leq n$. Mas para !Pedro, um segmento de um vetor é $k - bonito$ se o menor elemento desse vetor é menor ou igual a metade do intervalo, mais formalmente, se n é o menor número no intervalo de tamanho k, então $n \leq k/2$. Eles estão querendo saber, dado um vetor e um número k, quantos segmentos de tamanho k são $k - bonitos$ pela definição de Pedro e quantos são pela definição de !Pedro. Veja o primeiro caso de teste:

$[2, 3, 1, 5, 10, 9, 8]$

$k = 2$

A resposta seria 4 e 2, pois há 4 segmentos $k - bonitos$ para Pedro e há 2 segmentos bonitos para !Pedro.

Entrada

A entrada do programa consiste de uma linha contendo o nome do arquivo a ser lido, dentro do arquivo há apenas duas linhas, na primeira há o elemento k e na segunda são os elementos dos vetores, como descritos no enunciado (no arquivo não terá os colchetes e vírgulas, apenas os inteiros do vetor, separados por espaço). Todos os elementos do vetor são não negativos.

Saída

Imprima dois inteiros separados por espaço, o número de segmentos $k - bonitos$ de Pedro e os de !Pedro, respectivamente.

Notas

[illegible]do do arquivo usado no primeiro exemplo é exatamente o apresentado no enunciado.

[illegible]ple:

Aviso!

?

Input	Result
a.txt	4 2
b.txt	1 2
c.txt	3 3

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

```
filePath = input().strip()
qtdPedro, qtdFatorialPedro = 0, 0

with open(filePath, 'r', encoding='utf-8') as arquivo:
  content = arquivo.readlines()

k = int(content[0].strip())
vetor = list(map(int, content[1].strip().split(' ')))

def getSegmentos(vetor, tamanho):
  listaSegmentos = []
  for i in range(len(vetor)-tamanho+1):
    listaSegmentos.append(vetor[i:i+k])
  return listaSegmentos
```

	Input	Expected	Got	
✔	a.txt	4 2	4 2	✔
✔	b.txt	1 2	1 2	✔
✔	c.txt	3 3	3 3	✔

Aviso!

?

	Input	Expected	Got	
✔	d.txt	0 160	0 160	✔
✔	e.txt	0 121	0 121	✔
✔	f.txt	1007 2792	1007 2792	✔

Passou em todos os teste! ✔

Para resolver o problema, pode se passar por todos os segmentos de tamanho k, verificar o maior e o menor elemento e utilizar dois contadores para salvar quantos segmentos satisfazem a condição.

Question author's solution (Python3):

```
with open(input(), 'r') as f:
    linhas = f.read()
    linhas = linhas.split('\n')
    k = int(linhas[0])
    lista = [int(x) for x in linhas[1].split()]

    pedro, naoPedro = 0, 0
    for i in range(len(lista)-k+1):
        maximo, minimo = lista[i], lista[i]
        for j in range(i, i+k):
            maximo = max(maximo, lista[j])
            minimo = min(minimo, lista[j])
        if maximo >= 2*k:
            pedro += 1
        if minimo <= k//2:
```

Correto

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

Aviso!

Questão **9**

Correto

Atingiu 1,00 de 1,00

`Simple | Grep`

O comando (programa) **grep** é uma das ferramentas mais úteis e versáteis disponível no Unix. Ele procura por padrões especificados pelo usuário dentro de arquivos de texto. Em outras palavras, você pode pesquisar por palavras ou padrões e a linha ou linhas que possuem o conteúdo procurado serão exibidas.

Neste exercício, sua tarefa consiste em construir um programa para encontrar ocorrências de uma cadeia de caracteres em um arquivo, no estilo do programa *grep*.

Entrada

A entrada para seu programa é uma palavra p, um inteiro c e o nome n do arquivo, separados por um espaço entre cada um.

Saída

A saída do programa deverá ocorrer da seguinte forma: Se p aparece no arquivo, seu programa deve imprimir o nome do arquivo seguido do número da linha em que p aparece. Em seguida, c linhas anteriores a linha que contém p, a linha p e as c linhas posteriores a linha p. Entre uma ocorrência e outra, deve-se imprimir uma linha em branco, e se p não aparece em um certo arquivo, seu programa não deve imprimir nada.

Observações

- Cada linha dos arquivos de texto não tem mais do que 512 caracteres e a palavra e o arquivo não possuem mais do que 30 caracteres.
- No terceiro caso de teste, a palavra "Nao" está presente na linha 4 do arquivo estoria.txt. A saída então é "estoria.txt: 4". Na sequência é impressa 1 linha anterior ("na Praca Mayor. Pensas que se mataram?") a linha 4, a linha 4 ("Nao.") e 1 linha posterior a linha 4 ("Pensas que se feriram? Tambem nao. O desfecho depois").

For example:

	Result

Aviso!

?

Input	Result
vez 1 estoria.txt	estoria.txt: 1 Era uma vez dois toureiros que amavam a mesma mulher. Certo dia encontraram-se
Papa 2 gda.txt	
Nao 1 estoria.txt	estoria.txt: 4 na Praca Mayor. Pensas que se mataram? Nao. Pensas que se feriram? Tambem nao. O desfecho depois

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

```
palavra, qtdLinhas, filePath = input().strip().split(' ')
qtdLinhas = int(qtdLinhas)


with open(filePath, 'r', encoding='utf-8') as arquivo:
  texto = arquivo.readlines()

for i in range(len(texto)):
  if palavra in texto[i]:
    print(f'{filePath}: {i+1}')
    if i - qtdLinhas >= 0:
      begin = i - qtdLinhas
    else:
      begin = 0
    if i + qtdLinhas <= len(texto) - 1:
```

	Input	Expected	Got	

Aviso!

	Input	Expected	Got	
✔	vez 1 estoria.txt	estoria.txt: 1 Era uma vez dois toureiros que amavam a mesma mulher. Certo dia encontraram-se	estoria.txt: 1 Era uma vez dois toureiros que amavam a mesma mulher. Certo dia encontraram-se	✔
✔	Papa 2 gda.txt			✔
✔	perfeita 0 maquiavel.txt	maquiavel.txt: 1 Os romanos, entretanto, antevendo perfeitamente os inconvenientes	maquiavel.txt: 1 Os romanos, entretanto, antevendo perfeitamente os inconvenientes	✔
✔	Nao 1 estoria.txt	estoria.txt: 4 na Praca Mayor. Pensas que se mataram? Nao. Pensas que se feriram? Tambem nao. O desfecho depois	estoria.txt: 4 na Praca Mayor. Pensas que se mataram? Nao. Pensas que se feriram? Tambem nao. O desfecho depois	✔
✔	mal 2 maquiavel.txt	maquiavel.txt: 3 Os romanos, entretanto, antevendo perfeitamente os inconvenientes advindos desta incuria, sempre os remediaram e jamais permitiram a prorrogacao de tais males para furtar-se a uma guerra, pois sabiam que uma guerra nao evita-se mas protela-se, e nunca em seu proprio favor. Assim, preferiram guerrear contra Felipe e contra Antioco na maquiavel.txt: 11 o "gozar os beneficios do tempo", preferindo tirar proveito das suas proprias virtude e prudencia: sabiam que o tempo tudo arrasta consigo e que, assim, ele pode trazer o bem como o mal, o mal como o bem.	maquiavel.txt: 3 Os romanos, entretanto, antevendo perfeitamente os inconvenientes advindos desta incuria, sempre os remediaram e jamais permitiram a prorrogacao de tais males para furtar-se a uma guerra, pois sabiam que uma guerra nao evita-se mas protela-se, e nunca em seu proprio favor. Assim, preferiram guerrear contra Felipe e contra Antioco na maquiavel.txt: 11 o "gozar os beneficios do tempo", preferindo tirar proveito das suas proprias virtude e prudencia: sabiam que o tempo tudo arrasta consigo e que, assim, ele pode trazer o bem como o mal, o mal como o bem.	✔
✔	vez 1 gda.txt	gda.txt: 4 Minha melhor lembranca e esse instante no qual pela primeira vez me entrou pela retina tua silhueta provocante e fina	gda.txt: 4 Minha melhor lembranca e esse instante no qual pela primeira vez me entrou pela retina tua silhueta provocante e fina	✔

Aviso!

?

Passou em todos os teste! ✔

Para resolver a questão, recomenda-se armazenar todas as linhas do arquivo em uma lista. Em seguida, deve-se procurar em cada linha (que está em cada posição) da lista a palavra p. Se a palavra p existir em uma determinada linha/posição k da lista, deve-se utilizar uma estrutura de repetição para imprimir as c linhas anteriores à linha que contém p - tratando-se o caso em que a $k < c$. Utilize também uma estrutura de repetição para imprimir as c linhas posteriores a k, tratando-se o caso em que $k + c$ não ultrapasse o total de posições da lista.

Question author's solution (Python3):

```
word, c, file = input().split()

c = int(c)

f = open(file, "r")

lines = f.readlines()
cnt = len(lines)

for i in range(cnt):
    line = lines[i]
    if line.find(word) == -1:
        continue

    leftmost = max(0, i - c)
```

Correto

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

Aviso!

?

Questão **10**

Correto

Atingiu 1,00 de 1,00

Pesquisa de Preços

Você está interessado no menor preço para alguns equipamentos que estão em falta na sua empresa. Foi feita uma pesquisa de mercado em vários fornecedores diferentes, e você ficou responsável por dizer qual o melhor lugar para comprar cada item.

O resultado dessa pesquisa está em um arquivo do tipo **csv** (valores separados por vírgula - *comma-separated values*), formatado da seguinte maneira:

A primeira linha do arquivo é o cabeçalho. O primeiro campo do arquivo possui 2 números inteiros separados por espaço, n e m, que é a quantidade de fornecedores e a quantidade de itens a serem adquiridos, respectivamente. O resto da primeira linha é o nome dos n fornecedores.

As próximas m linhas do arquivo contém o nome do item no primeiro campo, seguido do preço desse item para cada um dos fornecedores.

Por exemplo, o arquivo pode ser:

```
4 3,CadeirasShop,MaterialParaEscritorio.com,Mercado da Esquina,Ármazem de usados
Cadeira,100.95,159.99,235.43,110.50
Mesa,330.00,220.55,440.39,115.34
Monitores,123.88,220.44,158.26,22.49
```

Dado o nome do arquivo, imprima o melhor fornecedor para cada item da lista.

Entrada

A entrada do programa consiste de uma linha contendo o nome do arquivo CSV a ser lido.

Saída

Imprima m linhas. Em cada linha imprima $item\ fornecedor$, separados por espaço. O fornecedor deve ser o que vende aquele item pelo menor preço.

Aviso!

- O primeiro exemplo utiliza o arquivo exemplo mostrado no enunciado.

For example:

Input	Result
1.csv	Cadeira CadeirasShop Mesa Ármazem de usados Monitores Ármazem de usados
2.csv	laranja CadeirasShop mouse Mercadinho fone de ouvido Escritorio.com mochila CadeirasShop calculadora CadeirasShop
3.csv	tomadas shopfy pasta de dente shopfy uniformes shopfy HDD Armazem de usados laranja kleber calculadora shopfy telefone kleber fone de ouvido shopfy

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

```
import csv

filePath = input().strip()

nomesFornecedores = []
itens = []

def getKey(valor, dicionario):
  for key in dicionario:
    if dicionario[key] == valor:
      return key
  return -1
```

Aviso!

```
12      return -1
13
14  with open(filePath, 'r', encoding='utf-8') as orcamentos:
15      leitor = csv.reader(orcamentos)
```

	Input	Expected	Got	
✔	1.csv	Cadeira CadeirasShop Mesa Ármazem de usados Monitores Ármazem de usados	Cadeira CadeirasShop Mesa Ármazem de usados Monitores Ármazem de usados	✔
✔	2.csv	laranja CadeirasShop mouse Mercadinho fone de ouvido Escritorio.com mochila CadeirasShop calculadora CadeirasShop	laranja CadeirasShop mouse Mercadinho fone de ouvido Escritorio.com mochila CadeirasShop calculadora CadeirasShop	✔
✔	3.csv	tomadas shopfy pasta de dente shopfy uniformes shopfy HDD Armazem de usados laranja kleber calculadora shopfy telefone kleber fone de ouvido shopfy	tomadas shopfy pasta de dente shopfy uniformes shopfy HDD Armazem de usados laranja kleber calculadora shopfy telefone kleber fone de ouvido shopfy	✔
✔	4.csv	mascara super saldos HDD Escritorio.com notebook melhor compra laranja fornecedor top mouse muy bueno	mascara super saldos HDD Escritorio.com notebook melhor compra laranja fornecedor top mouse muy bueno	✔
✔	5.csv	banana fun fun cadeira fun fun garrafa melhorCompra.com calculadora mallwow esponja bom d+	banana fun fun cadeira fun fun garrafa melhorCompra.com calculadora mallwow esponja bom d+	✔

Aviso!

?

	Input	Expected	Got	
✔	6.csv	laranja kleber jaleco eco-escritorios gaze bom d+ telefone eco-escritorios HDD melhores precos mochila batata mesa eco-escritorios banana super forncedor mouse eco-escritorios mascara kleber garrafa bom d+ monitor eco-escritorios	laranja kleber jaleco eco-escritorios gaze bom d+ telefone eco-escritorios HDD melhores precos mochila batata mesa eco-escritorios banana super forncedor mouse eco-escritorios mascara kleber garrafa bom d+ monitor eco-escritorios	✔

Passou em todos os teste! ✔

Esse é um exercício simples. O primeiro passo é separar a primeira linha do arquivo, que será usada como índice para o nome dos fornecedores. Lembrando que é necessário tirar a primeira célula do arquivo que contém o tamanho dele.

Após isso, basta escanear o arquivo linha a linha. Em uma linha l_i teremos primeiro o nome do item, e uma sequência de preços. Analisamos a sequência de preços para achar o menor, e pegamos o índice desse menor preço, idx_{menor}. Agora, vemos qual era o nome do fornecedor, que está em $primeira_linha[idx_{menor}]$.

Fazendo isso para todas as linhas resolve o problema

PS.: É necessário fazer o stripping da primeira linha para evitar imprimir uma quabra de linha extra se o menor preço vier do último forncedor.

Question author's solution (Python3):

```
1  csv = input()
2
3  def parse_file(lines):
4      header = lines[0].strip().split(',')
5      for line in lines[1:]:
6          item, *prices = line.split(',')
7          min_price = 1000000000
8          idx = -1
9          for i, p in enumerate(prices):
```

Aviso!

```
        for i, p in enumerate(prices):
            if float(p) < min_price:
                min_price = float(p)
                idx = i
        print(f'{item} {header[idx + 1]}')

with open(csv, 'r') as file:
```

Correto

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

Aviso!