Iniciado em	sábado, 18 set 2021, 18:57
Estado	Finalizada
Concluída em	domingo, 26 set 2021, 22:53
Tempo empregado	8 dias 3 horas
Avaliar	**8,86** de um máximo de 10,00(**89**%)

Aviso!

?

Questão 1

Parcialmente correto

Atingiu 0,29 de 1,00

Strings iguais

Dadas 2 strings a e b, você pode realizar 2 operações com elas:

1. Remover o primeiro caractere da string.
2. Remover o último caractere da string.

Repare que é possível que a string fique sem caracteres após alguma dessas operações.

Mostre o menor número de operações necessárias para deixar as duas strings iguais.

Entrada

A entrada consiste em 2 strings a e b, $1 \leq |a|, |b| \leq 50$.

Saída

O menor número de operações necessárias para deixar as duas strings iguais.

Observações

- No primeiro exemplo as duas strings já são iguais, então precisamos de 0 operações.
- No segundo exemplo temos que remover "ab" da primeira string. Independente se remover o primeiro ou o segundo "ab", são 2 operações.
- No terceiro exemplo realizamos 3 operações na primeira string e 1 na segunda. Dessa forma as duas se tornam a string vazia, e portanto iguais.

For example:

Input	Result
a a	0

Aviso!

Input	Result
xyz a	4

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

```
def get_maior_menor(a, b):
  if len(a) > len(b):
    maior = a
    menor = b
  else:
    maior = b
    menor = a
  return [maior, menor]

def chars_comuns(maior, menor):
  chars_comuns = []
  for x in menor:
    if maior.count(x) > 0:
      chars_comuns.append(x)
  return chars_comuns
```

	Input	Expected	Got	
✔	a a	0	0	✔
✔	abab ab	2	2	✔
✔	xyz a	4	4	✔

Aviso!

	Input	Expected	Got	
✔	aaaabcccc abc	6	6	✔
✔	aaaabc abc	3	3	✔
✖	hello helo	3	0	✖

Some hidden test cases failed, too.

Show differences

Primeiramente, existem soluções mais eficientes do que a que será apresentada aqui.

Esse problema basicamente requer que encontremos a maior **substring** que é comum entre as duas strings. Essa pode ser a string vazia.

Uma substring é uma sequência de caracteres contida na string.

Então uma solução relativamente óbvia, e que funciona **porque a entrada do problema é pequena** é testar todas as possíveis substrings de uma das strings na outra. Se encontrarmos uma substring comum, então o número de operações necessárias para deixar as duas strings igual à essa substring comum é $|a| + |b| - 2 * |substring|$.

Question author's solution (Python3):

```
def answer(a, b):
    for i in range(len(a)+1, 0, -1):
        for k in range(len(a)):
            if (k + i) > len(a):
                continue
            sub = a[k:k+i]
            if b.find(sub) != -1:
                return len(a) - len(sub) + len(b) - len(sub)
    return len(a) + len(b)

= input()
= input()
```

Aviso!

```
13 print(answer(a, b))
14
15 """
```

Parcialmente correto

Notas para este envio: 0,29/1,00.

Aviso!

?

Questão **2**

Correto

Atingiu 1,00 de 1,00

Está dentro?

Construa um programa para verificar, à partir de dois valores **A** e **B**, se **B** corresponde aos últimos dígitos (os menos significativos, mais à direita) de **A**.

Entrada

A entrada consiste de dois valores **A** e **B** maiores que zero, cada um deles podendo ter até 100 dígitos.

Saída

Imprima "ta dentro!!!" se **B** corresponde aos últimos dígitos de **A**, e "ta fora..." caso contrário

For example:

Input	Result
562345234857238547554545454786 90 78690	ta dentro!!!
5434554 543	ta fora...
1243 1243	ta dentro!!!

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

```
a, b = input().split(' ')

tamanho = len(b)
a_cortada = a[-len(b):]

if a[-len(b):] == b:
    print('ta dentro!!!')
else:
    print('ta fora...')
```

Aviso!

```
print( 'ta fora...' )
```

	Input	Expected	Got	
✔	562345234857238547554545454786 90 78690	ta dentro!!!	ta dentro!!!	✔
✔	5434554 543	ta fora...	ta fora...	✔
✔	1243 1243	ta dentro!!!	ta dentro!!!	✔
✔	54 64545454545454545454545454545454554	ta fora...	ta fora...	✔
✔	1234123 123	ta dentro!!!	ta dentro!!!	✔
✔	123123 2123	ta fora...	ta fora...	✔

Passou em todos os teste! ✔

A questão pode ser resolvida utilizando fatiamento da string A iniciando no final da string em direção ao início que considerando o tamanho da string b. Também pode-se utilizar funções buit-in do Interpretador Python que se aplicam a strings para verificar se a string A termina com o mesmo conjunto de caracteres presentes na string B. Por fim, também é possível resolver a questão percorrendo cada caractere da string B e cada caractere da sub-string de A que corresponde aos últimos n (n = tamanho da string B) caracteres de A.

Question author's solution (Python3):

Aviso!

```
B = list(map(str,input().split()))
if A[-len(B):] == B:
```

```
    print("ta dentro!!!")
else:
    print("ta fora...")

# Solução alternativa: Jeremias

# a, b = input().split()
# if a.endswith(b):
```

Correto

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

Aviso!

?

Questão 3

Correto

Atingiu 1,00 de 1,00

Escrita Polida

Depois de passar muito tempo na internet e perceber que as pessoas utilizam caixa alta para poder representar xingamentos e gritarias, lhe foi pedido que implementasse um programa que fosse exatamente na direção oposta a essas vulgaridades, que é transformar frases (algumas grotescas) em frases polidas. Para este caso, uma frase polida é aquela em que o primeiro caractere (que sempre é uma letra) está em maiúsculo e o restante das letras está em minúsculo.

Entrada
A primeira linha apresenta uma string de até 100 caracteres composta por caracteres legíveis e espaço.

Saída
Apresente, em uma linha, a frase constituída pelo primeiro caractere que é uma letra em maiúsculo e todas as outras letras em minúsculo.

For example:

Input	Result
`brasil`	`Brasil`
`unb`	`Unb`
`frase de teste.`	`Frase de teste.`

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

```
entrada = input()

print(entrada[0].upper() + entrada[1:].lower())
```

Aviso!

	Input	Expected	Got	
✔	brasil	Brasil	Brasil	✔
✔	unb	Unb	Unb	✔
✔	frase de teste.	Frase de teste.	Frase de teste.	✔
✔	se estiver lendo essa mensagem, levante a mao esquerda.	Se estiver lendo essa mensagem, levante a mao esquerda.	Se estiver lendo essa mensagem, levante a mao esquerda.	✔
✔	UMA FRASE QUALQUER, PARA VOCE TESTAR!	Uma frase qualquer, para voce testar!	Uma frase qualquer, para voce testar!	✔
✔	E SE A FRASE TIVER RETICENCIAS? COMO LIDAR...	E se a frase tiver reticencias? como lidar...	E se a frase tiver reticencias? como lidar...	✔

Passou em todos os teste! ✔

A questão pode ser respondida utilizando funções built-in do Interpretador Python para forçar um caractere para maiúsculo ou minúsculo. Você pode utilizar fatiamento para acessar o primeiro caractere e os demais.

Question author's solution (Python3):

Aviso!

```
= input()
```

```
a = x[0].upper()+x[1:].lower()
print(a)

# Solucao alternativa: Vinicius
#a = input()

#b = str()

#if ord(a[0]) > 96:
#  b=b+chr(ord(a[0])-32)
#else:
#  b=b+a[0]

#for i in range(1,len(a)):
```

Correto

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

Aviso!

?

Questão **4**

Parcialmente correto

Atingiu 0,86 de 1,00

Futebol

Nunão e Levinho amam futebol e estavam assistindo um clássico brasileiro, Corinthians x São Paulo. Estudiosos como são, eles escreviam as posições dos jogadores em uma peça de papel. Para simplificar a situação, as posições eram descritas como uma string contendo apenas 0's e 1's. Um "0" corresponde a um jogador do Corinthians e um "1" corresponde a um jogador do São Paulo.

Se existem 7 jogadores de um time, um do lado do outro, então a situação é considerada favorável àquele time. Por exemplo, a situação "00100110111111101" é favorável ao São Paulo e a situação "11110111011101" não é favorável a ninguém.

Determine para qual time a situação é favorável.

Entrada

A primeira linha de entrada contém uma palavra s - uma palavra com apenas "0"s e "1s" com comprimento entre 1 e 100. Existe pelo menos um jogador de cada time no campo.

Saída

Imprima "JOGO PESADO" se a situação for favorável para os dois times, "VAI TIMAO" se a situação for favorável apenas para o Corinthians, "VAMO TRICOLOR" se a situação for favorável apenas para o São Paulo e "BORA UM VIRTUAL NO CODEFORCES" se o jogo estiver entediante e a situação não for favorável para ninguém.

Notas

- No primeiro exemplo de teste, não existem 7 jogadores em sequência de nenhum dos dois times no campo (string da entrada);
- No segundo exemplo de teste, existem 7 jogadores do Corinthians em sequência (veja a sequência de zeros na strings), logo, a vantagem é do Corinthians;

For example:

Input	Result
	BORA UM VIRTUAL NO CODEFORCES

Aviso!

?

Input	Result
1000000001	VAI TIMAO
00100110111111101	VAMO TRICOLOR

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

```
s = input()
FAVORAVEL = 7
TIMAO = "0"
TRICOLOR = "1"


if FAVORAVEL*TIMAO in s and FAVORAVEL*TRICOLOR not in s:
  print("VAI TIMAO")
elif FAVORAVEL*TRICOLOR in s and FAVORAVEL*TIMAO not in s:
  print("VAMO TRICOLOR")
else:
  print("BORA UM VIRTUAL NO CODEFORCES")
```

	Input	Expected	Got	
✔	001001	BORA UM VIRTUAL NO CODEFORCES	BORA UM VIRTUAL NO CODEFORCES	✔
✔	1000000001	VAI TIMAO	VAI TIMAO	✔
✔	00100110111111101	VAMO TRICOLOR	VAMO TRICOLOR	✔
✔	110111111111111	VAMO TRICOLOR	VAMO TRICOLOR	✔

Aviso!

?

	Input	Expected	Got	
✔	01	BORA UM VIRTUAL NO CODEFORCES	BORA UM VIRTUAL NO CODEFORCES	✔
✔	1010010100000000010	VAI TIMAO	VAI TIMAO	✔

Your code failed one or more hidden tests.

Para resolver o problema, deve-se percorrer a string, caractere por caractere, e procurar por blocos de, no mínimo, sete caracteres "0" ou "1" consecutivos. Assim, ao percorrer a string, deve-se utilizar as variáveis para armazenar essas informações sobre os jogadores do São Paulo e do Corinthians.

Question author's solution (Python3):

```
s = input()

cnt = [0, 0]
f = [False, False]

prev = -1
for c in s:
    cur = ord(c)-ord('0')
    if cur != prev:
        cnt = [0, 0]

    prev = cur
    cnt[cur] += 1

    for i in range(2):
```

Parcialmente correto

Notas para este envio: 0,86/1,00.

Aviso!

?

Questão 5

Correto

Atingiu 1,00 de 1,00

Tesouro

Péricles está em uma caverna, onde supostamente existe um tesouro. No entanto está muito escuro na caverna e, portanto, Péricles pediu sua ajuda para saber quantos passos ele tem que andar até o tesouro, se é que existe um. Entretanto, deve-se considerar um problema adicional: partes da caverna colapsaram, então Péricles talvez não consiga chegar até o tesouro, mesmo que ele exista.

O mapa funciona da seguinte maneira: é uma string (a caverna é bem estreita, e só da pra andar para a esquerda ou direita) composta por ' ', '#', 'P' e 'X'. O espaço em branco ' ' indica uma parte intacta da caverna, por onde Péricles pode passar; 'P' é a posição de Péricles; 'X' indica o tesouro; e '#' indica uma parte da caverna colapsada, por onde Péricles não pode passar.

Entrada

A entrada consiste numa string s, indicando o mapa.

Saída

Imprima "Péricles, não tem tesouro" se não houver tesouro na caverna.

Imprima "Péricles esse caminho não funciona" se não houver um caminho que leve Péricles até o tesouro de sua posição atual.

Imprima "Péricles, d passos", onde d é o número de passos que Péricles precisa andar até o tesouro. Se for para a direita, d será um número positivo. Se for para a esquerda, d será um número negativo.

Observações

- No primeiro exemplo, Péricles está a 4 passos de distância do tesouro, e não tem nenhuma parede entre ele e o tesouro, então é possível chegar até lá;
- O segundo exemplo é semelhante ao primeiro, mas Péricles tem que se mexer para a esquerda, resultando em -4 passos;
- No terceiro exemplo de teste, existem partes da caverna colapsadas entre Péricles e o tesouro, então não é possível chegar até ele.

For example:

Input	Result
#	Péricles, 4 passos

Aviso!

?

Input	Result
#X P#	Péricles, -4 passos
# P ## X #	Péricles esse caminho não funciona

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

```
s = input()

def findall(string, character):
  indexes = []
  for i in range(len(string)):
    if string[i] == character:
      indexes.append(i)
  return indexes

def caminhoPossivel(tesouro, pericles, barreiras):
  for i in barreiras:
    if pericles < i and tesouro > i or pericles > i and tesouro < i:
      return False
  return True
```

	Input	Expected	Got	
✔	# P X#	Péricles, 4 passos	Péricles, 4 passos	✔
✔	#X P#	Péricles, -4 passos	Péricles, -4 passos	✔
✔	# P ## X #	Péricles esse caminho não funciona	Péricles esse caminho não funciona	✔
	P #	Péricles, não tem tesouro	Péricles, não tem tesouro	✔

Aviso!

	Input	Expected	Got	
✔	`#### P #X## ### # ### #`	`Péricles esse caminho não funciona`	`Péricles esse caminho não funciona`	✔
✔	`## ## #P X ## ### # # # #`	`Péricles, 15 passos`	`Péricles, 15 passos`	✔

Passou em todos os teste! ✔

O primeiro passo em resolver esse problema é encontrar a posição de Péricles e do tesouro no mapa. A função $find()$ do Python faz isso.

O segundo passo é ver se tem um tesouro ou não. Se não tiver um 'X' na string, já retornamos que não existe tesouro.

Caso ele existe, e sabendo a posição de péricles e do tesouro, ainda é necessário percorrer a string entre P e X (que podem estar invertidos) e ver se tem algum '#' entre eles. Caso tenha '#' no caminho, Péricles não poderá segui-lo.

Se o caminho estiver livre, Péricles precisa andar $P - X$ passos, onde P e X são as posições de Péricles e do tesouro, respectivamente.

Question author's solution (Python3):

```python
def caminho(s: str):
    p = s.find('P')
    x = s.find('X')
    if x == -1:
        return 'Péricles, não tem tesouro'
    L = min(p, x)
    R = max(p, x)
    for v in s[L:R]:
        if v == '#':
            return 'Péricles esse caminho não funciona'
    return f'Péricles, {x - p} passos'

```

Aviso!

este envio: 1,00/1,00.

Aviso!

?

Questão **6**

Correto

Atingiu 1,00 de 1,00

Split

Há um certo tempo, quando esta matéria ainda era dada em C, uma questão bastante clássica sobre strings era separar uma string em palavras. Os espaços marcam a separação entre duas palavras, é claro. Por exemplo a string `"atchim espirro banana maça"` seria separada em `"atchim", "espirro", "banana", "maça"`

Talvez nesse momento você esteja pensando que se a questão for só isso é muito fácil, porque Python convenientemente oferece a função `split()` que faz esse trabalho para você? Poisé, C não tinha essa função, e o ponto da questão é exatamente implementá-la. Como Python já tem um split, você terá que implementar um que seja um pouquinho mais poderoso.

Entrada

A entrada consiste em uma string s, composta por qualquer caractere exceto quebra de linha '\n'. **As palavras são sequências de letras** e elas estão separadas por **qualquer** caractere que não seja uma **letra**.

Os caracteres que separam palavras são diferentes entre si.

Cada palavra tem pelo menos 1 caractere.

Saída

Imprima cada uma das palavras contidas em s, uma por linha.

Observações

- No primeiro exemplo de teste, a string é "atchim espirro#banana2maça". Temos 4 palavras. As duas primeiras estão separadas por espaço. As próximas duas por '#', e as últimas duas por '2'. Imprimimos uma por linha.

For example:

Input	Result

Aviso!

Input	Result
atchim espirro#banana2maça	atchim espirro banana maça
Portugues:espirro]a9mano*distancia	Portugues espirro a mano distancia
abracadabra&complicado~APC=virgula,frita	abracadabra complicado APC virgula frita

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

```
def isLetter(character):
  if character.lower() in 'aáàãbcçdeéèfghiíìjklmnoóòõpqrstuvxywz':
    return True
  else:
    return False

s = input()
words = []
indexes = [-1]

for i in range(len(s)):
  if not isLetter(s[i]):
    indexes.append(i) # pegar os indexes dos caracteres não letras e fazer um split nesse
```

Aviso!

?

	Input	Expected	Got	
✔	atchim espirro#banana2maça	atchim espirro banana maça	atchim espirro banana maça	✔
✔	Portugues:espirro]a9mano*distancia	Portugues espirro a mano distancia	Portugues espirro a mano distancia	✔
✔	abracadabra&complicado~APC=virgula,frita	abracadabra complicado APC virgula frita	abracadabra complicado APC virgula frita	✔
✔	mano"estranho8musica!papagaio)legume,cadeira[onubus9acento	mano estranho musica papagaio legume cadeira onubus acento	mano estranho musica papagaio legume cadeira onubus acento	✔
✔	odsjfosdj!okefowekofewk5cheiro2facil frita$Python papagaio#abc	odsjfosdj okefowekofewk cheiro facil frita Python papagaio abc	odsjfosdj okefowekofewk cheiro facil frita Python papagaio abc	✔

Aviso!

?

	Input	Expected	Got	
✔	odsjfosdj!jogo1cadeira#estranho8da7nota@da%espaço=banana9facil	odsjfosdj jogo cadeira estranho da nota da espaço banana facil	odsjfosdj jogo cadeira estranho da nota da espaço banana facil	✔

Passou em todos os teste! ✔

O segredo da questão é que palavras são sequências de letras. Python oferece a função $isalpha()$ que diz se um caractere é uma letra ou não.
ENtçao basta manter 2 índices: i e j.

i marca o primeiro caractere de uma palavra, e j avança até encontrar um caractere que não é uma letra. Esse é o separador. Nesse ponto podemos imprimir $s[i:j]$, que é uma das palvras, e colocar i na posição $j+1$, que é o primeiro caractere após o separador, potencialmente o começo da próxima palavra.

Question author's solution (Python3):

```
1  def split(s):
2      i = 0
3      while i < len(s):
4          j = i + 1
5          while j < len(s) and s[j].isalpha():
6              j += 1
7          print(s[i:j])
8          i = j+1
9
10 s = input()
   split(s)
```

Aviso!

```
13 """
14 Uma implementação alternativa
15
```

Correto

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

Aviso!

?

Questão 7

Correto

Atingiu 0,86 de 1,00

Genoma

Semana passada a tumba do lendário feiticeiro Mellin foi aberta para que cientistas ao redor do mundo decodificassem seu DNA.

Essa tarefa está quase acabando, só o que falta é recuperar alguns nucleotídeos de uma cadeia s. Cada nucleotídeo é codificado como uma letra maiúscula do alfabeto português: "A", "C", "G" ou "T". Nucleotídeos não reconhecidos são codificados com um ponto de interrogação "?". Assim, s é uma cadeia com caracteres "A", "C", "G", "T" e "?".

Sabe-se que em uma cadeia, cada nucleotídeo deve aparecer um número igual de vezes.

Sua tarefa é decodificar o genoma e substituir cada nucleotídeo não-reconhecido por um dos quatro tipos reconhecidos de modo que o número de cada um dos quatro tipos seja igual.

Entrada

A primeira linha contém um inteiro n ($4 \leq n \leq 255$), representando o comprimento do genoma.
A segunda linha contém a string s de comprimento n - o genoma codificado, que é caracterizada por uma cadeia com caracteres "A", "C", "G", "T" e "?".

Saída

Se for possível decodificar o genoma, imprima "Segredos desvendados". Se não for possível, imprima "Feiticeiro misterioso" .

Notas

- No primeiro exemplo de teste, os pontos de interrogação do genoma podem ser substituídos por um caractere 'A', um caractere 'T' e um caractere 'G', em qualquer ordem;
- No segundo exemplo de teste, o genoma é válido;
- No terceiro exemplo de teste, não é possível obter uma quantidade igual de caracteres 'A', T', 'G' e 'C' ao substituir os caracteres possíveis nas posições dos pontos de interrogação no genoma.

Aviso! ple:

Input	Result
8 AG?C??CT	Segredos desvendados
4 AGCT	Segredos desvendados
6 ????G?	Feiticeiro misterioso

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

```
1  comprimento = int(input())
2  genoma = input()
3
4  #A quantidade de cada nucleotídeo A, C, G, T deve ser igual.
5
6  qtd_nucleotideos = [genoma.count('A'), genoma.count('C'), genoma.count('G'), genoma.count
7  qtd_desconhecida = genoma.count('?')
8
9  qtd_nucleotideos.sort(reverse=True)
10 if comprimento < 8 and qtd_nucleotideos[0]*4 == comprimento:
11   print('Segredos desvendados')
12 elif comprimento >= 8 and comprimento % 4 ==0:
13   print('Segredos desvendados')
14 elif qtd_desconhecida == comprimento and comprimento % 4 == 0:
15
```

	Input	Expected	Got	
✔	8 AG?C??CT	Segredos desvendados	Segredos desvendados	✔

Aviso!

?

	Input	Expected	Got	
✔	4 AGCT	Segredos desvendados	Segredos desvendados	✔
✔	6 ????G?	Feiticeiro misterioso	Feiticeiro misterioso	✔
✔	4 AA??	Feiticeiro misterioso	Feiticeiro misterioso	✔
✔	4 ????	Segredos desvendados	Segredos desvendados	✔
✔	252 ???????GCG??T??TT?????T?C???C?CCG???GA???????AC??A???AAC?C?CC??CCC??A??TA?CCC??T???C?? CA???CA??G????C?C?C????C??C??A???C?T????C??ACGC??CC?A?????A??CC?C??C?CCG?C??C??A??CG? A?????A?CT???CC????CCC?CATC?G??????????A?????????????????TCCCC?C?CA??AC??GC????????	Segredos desvendados	Segredos desvendados	✔

Passou em todos os teste! ✔

Para resolver o problema, leve em consideração que cada possível genoma, ou caractere ('A', 'T', 'G' e 'C'), tem exatamente a mesma quantidade: $n/4$. Assim, deve-se contabilizar cada caractere na string de entrada e verificar se a quantidade de pontos de interrogação ('?') são capazes de completar o genoma, de forma a manter a quantidade equilibrada de caracteres.

Question author's solution (Python3):

```
n = int(input())
s = input()


if (n % 4) != 0:
    print("Feiticeiro misterioso")
    exit()

mx = n // 4

nt_a = 0
nt_c = 0
```

Aviso!

```
13 cnt_g = 0
14 cnt_t = 0
15
```

Correto

Notas para o envio: 1,00/1,00. De acordo com as tentativas anteriores 0,86/1,00.

Aviso!

?

Questão 8

Correto

Atingiu 1,00 de 1,00

Consultas de intervalo na string 1

Guilherme está entediado nessa quarentena e começa a brincar com a criação de substrings com a ordem de seus caracteres invertidos, de tamanhos variados, a partir de uma string inicial. Sabendo disso, você entra em contato com Guilherme e faz q consultas sobre as possíveis substrings que podem ser geradas. Cada consulta contém dois números inteiros l e r e uma string $s = s_0 s_1 \ldots s_{n-1}$, em que $0 \leq l \leq r < n$ e s_i é um caractere alfanumérico. Para responder cada consulta, Guilherme deve gerar uma substring cujos caracteres estejam no intervalo $[l, r]$ da string s, isto é, $s_l s_{l+1} \ldots s_{r-1} s_r$, mas que sejam impressos em ordem inversa, obtendo-se $s_r s_{r-1} \ldots s_{l-1} s_l$.

Sua tarefa consiste em gerar a resposta do Guilherme para cada consulta.

Entrada

A primeira linha da entrada contém um número inteiro q ($1 \leq q \leq 100$), indicando a quantidade de consultas. As próximas q linhas descrevem todas essas consultas. Cada consulta é composta por dois números inteiros l e r, e por uma string s, nessa ordem e separados por espaço em branco. A string $s = s_0 s_1 \ldots s_{n-1}$, em que $1 \leq n \leq 100$, possui apenas caracteres alfanuméricos, e $0 \leq l \leq r < n$.

Saída

Para cada consulta, imprima uma única linha - a substring resultante $s_l s_{l+1} \ldots s_{r+1} s_r$ com seus caracteres em ordem inversa, isto é, deve-se imprimir a sequência de caracteres $s_r s_{r-1} \ldots s_{l-1} s_l$.

Observações:

- Não é necessário verificar se as relações entre l, r e n estão corretas. Assuma que o usuário digitou valores válidos.
- No primeiro caso de teste, devem ser realizadas duas consultas. Na primeira são selecionados os caracteres da posição 0 a posição 5 da string apresentada (Atleti) cuja ordem inversa é iteltA. Na segunda consulta, são selecionados os caracteres da posição 0 até a posição 6 da string apresentada (Athleti) cuja ordem inversa é itelhtA.

Aviso!

?

For example:

Input	Result
2 0 5 AtleticoGoianiense 0 6 AthleticoParanaense	iteltA itelhtA
3 0 9 TobiasBoon1995 10 16 noiprocSsVoreZbuS 0 2 xaJhitsHard	nooBsaiboT SubZero Jax
5 3 7 no123456Pwe 10 11 bf928f8wnfnwfjsijg 4 8 123456789FEDCBA 2 6 arararararara 0 0 v	65432 wn 98765 arara v

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

```
qtd_consultas = int(input())

for x in range(qtd_consultas):
  inicio, fim, texto = input().split(' ')
  inicio = int(inicio)
  fim = int(fim)
  string_cortada = texto[inicio:fim+1][::-1]
  print(string_cortada)
```

Aviso!

?

	Input	Expected	Got
✔	2 0 5 AtleticoGoianiense 0 6 AthleticoParanaense	iteltA itelhtA	iteltA itelhtA
✔	3 0 9 TobiasBoon1995 10 16 noiprocSsVoreZbuS 0 2 xaJhitsHard	nooBsaiboT SubZero Jax	nooBsaiboT SubZero Jax
✔	5 3 7 no123456Pwe 10 11 bf928f8wnfnwfjsijg 4 8 123456789FEDCBA 2 6 arararararara 0 0 v	65432 wn 98765 arara v	65432 wn 98765 arara v
✔	19 15 38 SqLkChItHcDeZsAeFlDdFlGtIxBnAgKqRzEsCkRiH 23 23 LfPsPkQvIfBmNqWiDbKzQiYgWfUnEfIwPxUeUmDwUgBfGnOjJrJpAf 8 33 RrUeOeZaBcQlZmCdWmKlVyZvUuGhEtWkXzUzF 13 25 RaReYbLwPrDnEtKyIgXyQpZxEcKyGyXtHsZx 45 45 DgLdNlZeQnDbVaCjQyHfKqVlIuSqMxYeQyQoRzMmJwSuIcNyXq 14 15 YlAqNvBjLlTeXgRhIfDnEuGySzNcRwGiDcF 6 23 ZkOqFqBwEuChTdVpIuNkPeHcShTmReNtGsNdNbTb 36 40 OgUoPyPwKaPkRwIsAmNxAgZfQsFsXtYxEpBpUmTlUjGxUeNmZg 31 31 GiUtQoElTnLyBoQjEgCvUuIiLmSbNaLvB 26 26 TyArEnKeWzLpWgQzFnNkExXdSuFcJv 15 39 EsRoBoSuZuMuCcGmRsMxZtPsIqChFcLvZkMvFmUsCnBrLcZvFzWmMkTuSaJd 34 55 MpRjLnDyDxRoJjAhOcItArXlQxQjXqHpFzEwPyYkZeQjPeHsGiQvYeZbQwN 12 28 CiObBlOxJxUoZsUvQwPhLhGlUfBhJgB 1 7 JxIyEuWmDuSfMlXkVqSmWyTkJo 20 27 NpGpHeFwPqNrBxWdAiPpUeOlNgEdDtRxPaXxZiWpHx 32 40 NqSdIvGpLbCsZjHsXeIcKsXbSeJhGmKiHtJcQqWxFtJsWwPtM 1 20 TqNoIxWpArKlAyJdSbIjTrGtXgZfCpUcHdSmKvQrHg 14 14 DnNuNwSxScQqNnQmPcPkAgZsXmCbOrWjYqNnOuS 32 32 DmKlMnDnTfUkMsKhNfJnFrVhQoMoFzKqIpTs	RkCsEzRqKgAnBxItGlFdDlFe g zXkWtEhGuUvZyVlKmWdCmZlQcB cExZpQyXgIyKt c hR cHePkNuIpVdThCuEwB UlTmU v F sUmFvMkZvLcFhCqIsPtZxMsRm bZeYvQiGsHePjQeZkYyPwE JhBfUlGhLhPwQvUsZ mWuEyIx dEgNlOeU FxWqQcJtH TjIbSdJyAlKrApWxIoNq Q I	RkCsEzRqKgAnBxIt g zXkWtEhGuUvZyVlK cExZpQyXgIyKt c hR cHePkNuIpVdThCuE UlTmU v F sUmFvMkZvLcFhCqI bZeYvQiGsHePjQeZ JhBfUlGhLhPwQvUs mWuEyIx dEgNlOeU FxWqQcJtH TjIbSdJyAlKrApWx Q I
		lTcRvTsTjFlF	lTcRvTsTjFlF

Aviso!

?

	Input	Expected	Got
✔	36 47 UjJxKbUrWsBdLlApWrQcArXmLzQwFcAlVwXjFlFjTsTvRcTlB	nUbGgPcWoIhZwBfDmTiCzRwEcEqWbPe	nUbGgPcWoIhZwBfD
	11 41 JpNxYhScXdXePbWqEcEwRzCiTmDfBwZhIoWcPgGbUnWiOpNjDjCwRm	EjAbRfZnVoRpSyQgKvHbMjDrWdLaId	EjAbRfZnVoRpSyQg
	3 32 QuLdIaLdWrDjMbHvKgQySpRoVnZfRbAjEsSmYbYcKdQmSxSrYdSk	KsYwTbMoJxF	KsYwTbMoJxF
	2 12 TyFxJoMbTwYsKcTdJfQuVe	uW	uW
	26 27 KrWiVzAyCtXcFxPtGrUpRmPnZsWuOeGwGdB	GrDgDsZzOqO	GrDgDsZzOqO
	18 28 IrUiSjQiBaCpBoMbSjOqOzZsDgDrGyQyR	uGcExZqRqFdUg	uGcExZqRqFdUg
	5 17 SsNtFgUdFqRqZxEcGuClNxEaTmLqXsJkYjXiFiRsWzUdOlG	gOtOqAwRsLhWcTg	gOtOqAwRsLhWcTg
	21 35 JjZjUjUaYjIiCcErVhAvVgTcWhLsRwAqOtOg	hE	hE
	22 23 TwGjMfQsLvHzPgNsFqHgBoEhMzPo	lZeUlN	lZeUlN
	10 15 KqJuAnPnUfNlUeZlJsQvToFmEnWf	tZoYiExNzCwWbQwPkBsNeGyLhUhRqBoGo	tZoYiExNzCwWbQwP
	7 39	FgDu	FgDu
	RrPzEtXoGoBqRhUhLyGeNsBkPwQbWwCzNxEiYoZtKmGcVjUsUrKtIeHaHzR	CfOeCjRoMhUtVdGoFvV	CfOeCjRoMhUtVdGo
	19 22 RaSiKbAbWcSrHaCzNxYuDgFpQdP	RzFmZ	RzFmZ
	6 24 JaWgHwVvFoGdVtUhMoRjCeOfCtZqYgFiEtZk	AeUhV	AeUhV
	4 8 KbSjZmFzRmFpMgVoAfXfYi	lXsZoUdS	lXsZoUdS
	16 20	lAoItZkPfC	lAoItZkPfC
	YuXcJrEdRyDnVlXwVhUeAiQgLuUfBbDtPvTcFhPrLyRbVvLnTrQzMqAwBsSr	XzUd	XzUd
	8 15 HaXtJvSxSdUoZsXlOeBlCiFbYf	cAkLfBy	cAkLfBy
	8 17 NsYrIcVcCfPkZtIoAlHqHqEkYdZqX	ZsRwIfFgCiUgJyIsFg	ZsRwIfFgCiUgJyIs
	33 36 CaPlZlYxVfBpDzMlSfMmLgTnFrArOeBtFdUzXgPsAqWcNjI	cCmLlCyRlQ	cCmLlCyRlQ
	21 27 RzYfWkRdLaByWhCmFgZvCyBfLkAcYhHcKaSmBlHbD	AxWrYzBsPaWzDmGv	AxWrYzBsPaWzDmGv
	15 32 JnFwYlCqVnBmZxUgFsIyJgUiCgFfIwRsZzDbZjYhVgPnFsApUf	sPyHxEwKrUwRuUkDcHoHtVrRxUqM	sPyHxEwKrUwRuUkD
	40 49 EsPxBfLjGtDgSmFeEcQfWfVkWeIaCdItMsNfAlDcQlRyClLmCcM	InRe	InRe
	5 20 NlBkWvGmDzWaPsBzYrWxAsQvCtBmTbPiDeDmVrPxQnDcAhLtG	HlKiI	HlKiI
	6 33 JaRuUeMqUxRrVtHoHcDkUuRwUrKwExHyPsBuPx	mCaMxUdSeHzIbWhC	mCaMxUdSeHzIbWhC
	33 36 DeShLtSvFhVjVhNxGgArVcDaIiAqAdYjJeRnId	nYpViPyEfEr	nYpViPyEfEr
	16 20 SjDqUcSaUpRzTyFoIiKlHtJsZnDdF	yRkYkG	yRkYkG
	2 17 ElChWbIzHeSdUxMaCmZiLdEcEgS	sSnSeXxXcIvZiEqHeNfZrCcEhYxDxBvBuCkWnLpAyJqV	sSnSeXxXcIvZiEqH
	45 55	aXvUiS	aXvUiS
	TfZnBbBgXaVmPnFlHpXyLxKyZtWbQwPuYmQlNxMeTgHtNrEfEyPiVpYn	Hs	Hs
	24 29 BrEsGvLmKkXwHlOmIgIrIzLnGkYkRyMhEyWuJwMjH	YmUnD	YmUnD
	12 55	yY	yY
	MgMkQgSaJkYkVqJyApLnWkCuBvBxDxYhEcCrZfNeHqEiZvIcXxXeSnSsFe	oKvUyWuPuJcJaWhYfNe	oKvUyWuPuJcJaWhY
	8 13 SjDhXnDdSiUvXaNfKpEqOdIrLsYwLyMiWsHvEnLiUoMfLsYiQaCj	DcAyPsYqAmJkC	DcAyPsYqAmJkC
	19 20 HxMeKlFaDfVxVmPkNeDsHb	oIjYbWyUq	oIjYbWyUq
	44 48 NbOqUeAaIqWeXnQgQvJwZfOfOkVkGdGzHwRtKtDbErApDnUmYt		
	14 15 BqCoZoJiVhKnGvYyStJgSdFnOsJ		
	3 21 PzFeNfYhWaJcJuPuWyUvKoRv		
	20 PlWvOoOjCkJmAqYsPyAcDnGdNmU		
	51 OiSlCmOrSfGzZkSpOsLtNxPvOhDsPmZkEwHwNjMmMgIqUyWbYjIoW		

Aviso!

?

	Input	Expected	Got
✔	17 22 29 FlTaJjRdUgEjYjRkWaZoGeRwKhVgTqUxUxHrUbFrQyFa 5 37 AePoRgWqUiJhBuWqVeJgCoJnCyUoPfGfPtTeUsXaJvQyN 21 25 SgNxJhJeToMwBtTpGbSzZuWrFnOrZxToMuWnQpKuTtCbYbDhLu 30 34 XtUoXmZgIbUlSzObTmXuTcClNyXcDfHzNmWhEsGeZpVxPyCzLcGyAqNk 4 45 UkIiIuCjFuEqBmXqNwDcYbErSzMoFrWoClByNjWjCrQaYjPc 14 57 DyZmCtZbAoQeUgWrMhPkZvYyGkOvPcKbZeTsHnEbUhTmQqPmBwSeOcXeOb 9 19 MkFmZvMkEzWoMlYzPaGrXgXyBzTs 0 26 UnEtDtIkJbBxUqJcDwYhWrEgFyJhCtU 25 28 QoFgWhPyUtWrByEbFsSwBrCzWxKlT 18 36 MfUxArYdAgGoJpHeUsIuBpMqJmHoCqMvJoSpDfLiVmSrCqTw 3 35 AzUnYtIfHxQuAsVcGdLoZuQzWhFsSrCxArCjFlFtMwNgZoNwFsVtUkXvBs 20 21 BbPbFwJfNgElEkPmOaDiZdSoRtKaBmSwCmYwNbAj 49 49 VmNdBvTcPnJxWmOdDnTsQtHuSvZmKuEdBdKbIqDwPwPsSrJjFlSwUfMh 31 33 NsTzUbTcVqVqSdSsGsCjYqZwCaMnVrIlEsSqPzGqXdLsNcL 27 56 SaQbXxYlTzLqWlUoRtKqAdJcQuQdQcSvIxGwGlWrRmKhDsBeBbJvCgZlW 1 22 AsPhVmNfVfJyKqMiMwXeMmAhHmZ 0 26 ApWoRuVlPqOlWbMkEsKaFyZbVzVmMpRwAxIsDuKvMbVcVzWxYnRtK	qTgVhKwR sUeTtPfGfPoUyCnJoCgJeVqWuBhJiUqWg nFrWu WmNzH jYaQrCjWjNyBlCoWrFoMzSrEbYcDwNqXmBqEuFjCuI bOeXcOeSwBmPqQmThUbEnHsTeZbKcPvOkGyYvZkPhMrW rGaPzYlMoWz JyFgErWhYwDcJqUxBbJkItDtEnU TlKx DpSoJvMqCoHmJqMpBuI jCrAxCrSsFhWzQuZoLdGcVsAuQxHfItYn dZ l sEl WlZgCvJbBeBsDhKmRrWlGwGxIvScQd AmMeXwMiMqKyJfVfNmVhPs VzVbZyFaKsEkMbWlOqPlVuRoWpA	qTgVhKwR sUeTtPfGfPoUyCnJ nFrWu WmNzH jYaQrCjWjNyBlCoW bOeXcOeSwBmPqQmT rGaPzYlMoWz JyFgErWhYwDcJqUx TlKx DpSoJvMqCoHmJqMp jCrAxCrSsFhWzQuZ dZ l sEl WlZgCvJbBeBsDhKm AmMeXwMiMqKyJfVf VzVbZyFaKsEkMbWl

Passou em todos os teste! ✔

A questão pode ser resolvida utilizando iteração para percorrer a string do final para o início. Outra forma de resolver a questão é realizar o fatiamento da sub-string de acordo com o solicitado e utilizar fatiamento para retornar a sub-string reversa.

Question author's solution (Python3):

```
q = int(input())

while q:
    l,r,string = input().split()
    l = int(l)
    r = int(r)
```

Aviso!

```
7   output = str()
8   for i in range(r,l-1,-1):
9     output=output+string[i]
```

Correto

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

Aviso!

?

Questão **9**

Correto

Atingiu 1,00 de 1,00

Apagando zeros

Você tem uma string s em que os caracteres são ou '0' ou '1'.

Você quer que todos os 1's da string formem uma sequência contínua. Por exemplo, se a string é 0, 1, 00111 ou 01111100, então todos os 1's formam uma sequência contínua. Se a string for 0101, 100001 ou 111111111101, então a condição não é satisfeita.

Você pode apagar alguns (ou nenhum, se quiser) 0's (zeros) da string. Qual o menor número de 0's que você tem que apagar para que a string da entrada tenha a condição satisfeita?

Entrada

A primeira linha contém um número n ($1 \leq n \leq 100$), indicando a quantidade de strings você terá que responder.

As próximas n linhas contém uma string s de tamanho entre 1 e 100 caracteres, em que cada caractere é ou 0 ou 1.

Saída

Imprima n inteiros, onde o i-ésimo inteiro é a resposta para a i-ésima string da entrada.

Observações

- No caso de teste 1, para que os 1's da string 010011 tenham sequência contínua, o mínimo de O's que precisam ser apagados é 2 que se encontram na posição 2 e 3 da string. A string 0 não possui caracter 1, portanto a resposta é 0. E a string 1111000 já possui todos os 1s em sequência contínua, então a resposta também é 0.
- No caso de teste 2, para que os 1's da string 01010 tenham sequência contínua, o mínimo de O's que precisam ser apagados é 1 que se encontra na posição 2 da string. A string 0 não possui caracter 1, portanto a resposta é 0.
- No caso de teste 3, para que os 1's da string 00101111100 tenham sequência contínua, o mínimo de O's que precisam ser apagados é 1 que se encontra na posição 3 da string.

For example:

Aviso!

Input	Result
3 010011 0 1111000	2 0 0
2 01010 0	1 0
1 00101111100	1

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

```
casos = int(input())
for x in range(casos):
  entrada = input()
  qtd_um = entrada.count('1')
  if qtd_um > 0:
    if '1'*qtd_um not in entrada:
      primeiro_um = entrada.index('1')
      ultimo_um = len(entrada) - 1 - entrada[::-1].index('1')
      qtd_apagados = entrada[primeiro_um:ultimo_um].count('0')
    else:
      qtd_apagados = 0
  else:
    qtd_apagados = 0
  print(qtd_apagados)
```

	Input	Expected	Got	

Aviso!

	Input	Expected	Got	
✔	3 010011 0 1111000	2 0 0	2 0 0	✔
✔	2 01010 0	1 0	1 0	✔
✔	1 00101111100	1	1	✔
✔	14 00101111011111111011100011010 10111000010111110010 00110001101101111101111010110110010110001001111100010011 10011111011110110111010011110101111101100100001111110011 10110001111011111111000110011111100100110010000100010000011100000100101111100010111 11110100001001100100000000000011111110101010110001100101001101010011000010010101100001001100111011 10010001110010101000111101011101110100110001111100110101001011010101101111101111000010001111100001 01100001001011010111001010010000000000101100011000101001101 10111110101100100100100001111010011101000011101111010101100001110001111110000001101101111110101000011 11100101001111111010101110011111101100001100000100010111011101011010000011101001001010001110110 0 01101111110110010000011010110001101010101100100110110010 10111101100111100110001100101 1 01111111100101100100111000110 01 10101000000110111000011010001110111111110011010110101 1	8 8 22 19 40 54 44 35 45 43 25 12 12 24	8 8 22 19 40 54 44 35 45 43 25 12 12 24	✔

Aviso!

?

	Input	Expected	Got	
✔	15 0010000101111010101001001110011111001011000100011110111100110010000 1 100101 0001011101000000101000110100100001101011010 1 11001111110110001101011 11011101110110101011010110010111100100000110 11010011001011001010011111010000001001110011100010000 1 011111100110111 00110001111011100110001011000111001111010000110011111011100100111111010101 11111100110000011010100011111011001011010110101101101110011111111000001100 10010010011011101110100111010011000100101100000010011010100100000101100111100001100101011111111001 11001110001000011110111100000010111101110 1 11110010011110001110100001010000100 1011 10000011100101101100101001100100000101000010101111010000 1 01011001010001001000110010111111111111 10101110111110111011010000000011111100101111	32 3 23 8 17 29 3 30 29 50 19 20 33 15 18	32 3 23 8 17 29 3 30 29 50 19 20 33 15 18	✔
✔	14 10100101000010101101000110100110111011100011000010100110111100101110101100111010001000 1 00111001010110011011111111110100101110101001110010100100001010111100111000110010111101 11100101101001010110011100010100110111100100110000 1 00110010001101101110001000110111010110111111001001011 00 00100110110100011011111 00001111111000000101001011011110010011101011101010 1 00111000010001110101011 0 10000111001101001110001000011100010010011001011011111110101011100001101101010001111001111111011101 10010111001110001101001000111011100100100111110010000011011000111100100 1010 11110110110101011110001101101010011110111011000010001010010100000001000101000000001110010 1010 11111111100001111000010000111000100011010010011110001000110100011111101010100 11011011 11011010101001111001111001111000110100001010101000000111111100100110100000 10 011100000110001 11010101000010110101010110101010101101110110101111100101010100100110001110100011001100010111001 1010	41 35 21 22 8 20 10 44 37 49 40 37 8 46	41 35 21 22 8 20 10 44 37 49 40 37 8 46	✔

Passou em todos os teste! ✔

Para resolver o problema, deve-se percorrer a string lida da entrada e contabilizar por todos os caracteres '0' que apareçam em posições da string após a primeira ocorrência do primeiro caractere '1' e antes da última ocorrência do caractere '1'.

Aviso!

Question author's solution (Python3):

?

```
n = int(input())

for i in range(n):
    s = input()

    stored = 0
    cnt = 0
    active = False

    for c in s:
        if c == '1':
            stored += cnt
            cnt = 0
            active = True
        elif active:
```

Correto

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

Aviso!

?

Questão 10

Parcialmente correto

Atingiu 0,86 de 1,00

Intervalo musical

Por agora, você já deve conhecer o DJ Brines certo? Não? Caso não o conheça, $esta$ é uma de suas músicas. Agora está na hora dele fazer mais músicas! Porém, Brines está em turnê pelo Brasil e não está tendo tempo de compor uma música nova. Assim, você, grande amigo e produtor musical, vai ajudá-lo com alguns arranjos musicais. Você sabe que para um certo intervalo na música, quanto mais notas intercaladas existirem, melhor o intervalo será. Então, seu trabalho será ler 2 strings s e t sendo as notas musicais e dois números n e m correspondendo a quantidade de vezes que cada uma das notas deve aparecer e computar qual o melhor intervalo.

Entrada

A entrada consiste em duas linhas. A primeira contém uma string s ($1 \leq |s| \leq 3$) e um número inteiro x ($1 \leq x \leq 100$) e a segunda uma string t ($1 \leq |t| \leq 3$) e um número inteiro y ($1 \leq y \leq 500$). Em cada linha a string representa a nota e o número indica a quantidade de vezes que ela deve aparecer.

Saída

Imprima uma string, que representa a sequência de notas para aquele intervalo.

Regra 1: Comece pela nota de maior frequência.

Regra 2: Se as notas tiverem a mesma quantidade, comece pela primeira nota.

Observações

- No primeiro exemplo de teste, o resultado é $fasolfasolfasolfasol$ pois é a única forma de maximizar as notas intercaladas que satisfaça ao problema.

For example:

Aviso!

esult

Input	Result
fa 4 sol 4	fasolfasolfasolfasol
la 13 si 5	lasilasilasilasilasilalalalalalalalala
do 20 re 18	doredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredodo

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

```
nota1, qtd1 = input().split()
nota2, qtd2 = input().split()

qtd1 = int(qtd1)
qtd2 = int(qtd2)

ORDEM = ['do', 're', 'mi', 'fa', 'sol', 'la', 'si']
if nota1 in ORDEM and nota2 in ORDEM:
  if ORDEM.index(nota1.lower()) < ORDEM.index(nota2.lower()):
    primeira = nota1
    segunda = nota2
  else:
    primeira = nota2
    segunda = nota1
else:
```

	Input	Expected
✔	fa 4 sol 4	fasolfasolfasolfasol
	13 5	lasilasilasilasilasilalalalalalalalala

Aviso!

?

	Input	Expected
✔	do 20 re 18	doredoredoredoredoredoredoredoredoredoredoredoredoredoredoredodo
✔	mi 150 si 149	misimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisimisim
✔	bri 59 nes 59	brinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbrinesbri
✔	do 100 re 50	doredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredoredored

Your code failed one or more hidden tests.

Para resolver o problema, deve-se intercalar as duas notas até a frequência da menor acabar e preencher o fim com o que sobrar.

Question author's solution (Python3):

```
def solve(s, n, t, m):
    n = int(n)
    m = int(m)
    if m > n:
        s, t = t, s
        n, m = m, n
    res = ''

    for i in range(n):
        res += s
        if i < m:
            res += t

    return res
```

Aviso!

```
14        return res
15    s, n = input().split()
```

Parcialmente correto

Notas para este envio: 0,86/1,00.