INSTITUTO DE COMPUTAÇÃO

Universidade Federal do Rio de Janeiro
Centro de Ciências Matemáticas e da Natureza

Computação 1 2026-2 (Eng. Produção)

Prova 1

23 de outubro de 2026

Justifique todas as suas respostas!

Boas práticas de programação (documentação, escolhas de bons nomes, etc.) fazem parte da avaliação. Você pode utilizar a função que responde uma questão da prova na resposta de qualquer outra (mesmo que não tenha resolvido a questão que está usando), desde que isso não crie dependências circulares.

Questão 1.

a (2 pontos). Faça uma função que receba duas `string`s, sendo a segunda de comprimento 1 (você pode assumir isso), e retorne a primeira `string` com a primeira ocorrência da segunda `string` (se existir) removida.

Exemplo: com argumentos `'casaco'` e `'a'`, o retorno deve ser `'csaco'`; com argumentos `'Hugo'` e `'x'`, o retorno deve ser `'Hugo'`.

b (2 pontos). Faça uma função que receba uma `string` não-vazia e retorne uma versão "embaralhada" da mesma `string`: os mesmos símbolos, nas mesmas quantidades, mas em ordem aleatória. *Dica*: O módulo `random` tem uma função `choice` que recebe uma sequência e retorna um elemento aleatório dela.

Exemplo: com argumento `'imaginedragons'` uma saída possível é `'raggedinsomnia'`.

c (2 pontos). Faça uma função para gerar senhas aleatórias: ela deve receber um inteiro e retornar uma `string`, com aquele comprimento, contendo:

- pelo menos uma letra minúscula
- pelo menos uma letra maiúscula
- pelo menos um dígito de `0` a `9`
- pelo menos um símbolo dentre as possibilidades `@#$%&`.

Caso o valor do inteiro dado não permita que isso possa ser feito, sua função deve retornar `False`.

Questão 2. Sejam n e k inteiros positivos. Duas crianças **A** e **B** estão fazendo uma brincadeira: elas recitam os números naturais (começando em 0), em ordem crescente, no ritmo de um número por segundo, mas com os seguintes detalhes:

- a cada n segundos, **A** volta a sua contagem a 0;
- a cada k segundos, **B** volta a sua contagem a 0.

Suponhamos que eles tenham começado a brincar exatamente *agora*, no "segundo 0", com ambos **A** e **B** falando "0". A brincadeira acaba quando ambos falarem "0" novamente ao mesmo tempo.

Por exemplo, se $n = 4$ e $k = 6$, a brincadeira fica assim:

segundo	**A** fala	**B** fala
0	0	0
1	1	1
2	2	2
3	3	3
4	0	4
5	1	5
6	2	0
7	3	1
8	0	2
9	1	3
10	2	4
11	3	5
12	0	0

a (2 pontos)**.** Faça uma função que receba `n` e `k` como argumentos e imprime ("`printa`") a brincadeira na tela, no formato que você desejar (pode ser como na tabela acima, mas não precisa – a escolha é sua). A impressão pode ser toda de uma vez, não precisa ser no ritmo da brincadeira onde as crianças falam uma vez por segundo.

b (2 pontos)**.** Faça uma função que receba `n` e `k` como argumentos e retorne um `bool` (`True` ou `False`) indicando se todos os pares x, y de inteiros não-negativos com $x < n$ e $y < k$ são tais que, em algum momento, **A** falará "x" no mesmo momento em que **B** falou "y".

(No exemplo acima, a resposta é `False`: por exemplo, falta o par "**A** fala 0, **B** fala 1".)

sequências em geral

- `seq.count(elem, início, limite)`

Retorna o número de vezes que `elem` ocorre em `seq[início:limite]`. Os argumentos `início` e `limite` são opcionais e têm a mesma interpretação que na notação de fatiamento de sequências.

- `seq.index(elem, início, limite)`

Retorna o menor índice de `seq` no qual `elem` ocorre em `seq[início:limite]` (dando `ValueError` caso não ocorra). Os argumentos `início` e `limite` são opcionais e têm a mesma interpretação que na notação de fatiamento de sequências.

tipo `string`

- `str.capitalize(string)`

Retorna a `string` com seu primeiro caracter em letra maiúscula, demais em letra minúscula.

- `str.center(string, largura, preench=' ')`

Retorna a `string` em versão centralizada, ocupando `largura` espaços no mínimo. O caracter `preench` é usado para preencher a largura, caso necessário.

- `str.endswith(string, sufixo, início, limite)(...)`

Retorna `True` se `string[início:limite]` termina exatamente como `sufixo`. Os argumentos `início` e `limite` são opcionais e têm a mesma interpretação que na notação de fatiamento de sequências.

- `str.find(string, substring, início, limite)`

Retorna o menor índice da `string` na qual a `substring` é encontrada, com a busca sendo feita apenas na fatia `string[início:limite]`. Os argumentos `início` e `limite` são opcionais e têm a mesma interpretação que na notação de fatiamento de sequências. Retorna `-1` caso a `substring` não seja encontrada.

- `str.isalnum(string)`

Retorna `True` se a `string` é não vazia e composta apenas por caracteres alfanuméricos (letras e dígitos).

- `str.isalpha(string)`

Retorna `True` se a `string` é não vazia e composta apenas por caracteres alfabéticos (letras).

- `str.isdigit(string)`

Retorna `True` se a `string` é não vazia e composta apenas por caracteres numéricos (dígitos).

- `str.islower(string)`

Retorna `True` se a `string` contém ao menos uma letra minúscula, e todas as suas letras são minúsculas.

- `str.isupper(string)`

Retorna `True` se a `string` contém ao menos uma letra maiúscula, e todas as suas letras são maiúsculas.

- `str.ljust(string, largura, preench=' ')`

Retorna a `string` em versão alinhada à esquerda, ocupando `largura` espaços no mínimo. O caracter `preench` é usado para preencher a largura, caso necessário.

- `str.lower(string)`

Retorna uma cópia da `string` com letras convertidas para minúsculas.

- `str.lstrip(string, chars=None)`

Retorna uma cópia da `string` com os elementos de `chars` removidos do início. Caso não seja fornecido o argumento `chars`, espaços em branco são removidos.

- `str.removeprefix(string, prefixo)`

Retorna uma cópia da `string` com o `prefixo` dado removido do início.

- `str.removesuffix(string, sufixo)`

Retorna uma cópia da `string` com o `sufixo` dado removido do fim.

- `str.replace(string, old, new, count=-1)`

Retorna uma cópia da `string` com as `count` primeiras ocorrências de old trocadas por new. Caso `count` seja `-1` (valor padrão), todas são trocadas.

- `str.rfind(string, substring, início, limite)`

Como `str.find`, mas buscando a partir do final pro começo da `string`.

- `str.rjust(string, largura, preench=' ')`

Retorna a `string` em versão alinhada à direita, ocupando `largura` espaços no mínimo. O caracter `preench` é usado para preencher a largura, caso necessário.

- `str.rstrip(string, chars=None)`

Como `str.lstrip`, mas removendo do final da `string`.

- `str.startswith(string, prefixo, início, limite)`

Retorna `True` se `string[início:limite]` começa exatamente como `prefixo`. Os argumentos `início` e `limite` são opcionais e têm a mesma interpretação que na notação de fatiamento de sequências.

- `str.strip(string, chars=None)`

Retorna uma cópia da `string` com os elementos de `chars` removidos do início e do fim. Caso não seja fornecido o argumento `chars`, espaços em branco são removidos.

- `str.swapcase(string)`

Troca as letras maiúsculas por minúsculas e vice versa em `string`.

- `str.title(string)`

Retorna uma cópia de `string` com a primeira letra de cada palavra convertida para maiúscula, e as demais para minúsculas.

- `str.upper(string)`

Retorna uma cópia da `string` com letras convertidas para maiúsculas.

módulo `math`

- `ceil(x)`

Retorna o "teto" de `x`, ou seja, o menor inteiro maior ou igual a `x`

- `floor(x)`

Retorna o "piso" de `x`, ou seja, o maior inteiro menor ou igual a `x`

- `gcd(*inteiros)`

Retorna o maior divisor comum (mdc) dos inteiros dados. A quantidade de argumentos é variável.

- `lcm(*inteiros)`

Retorna o menor múltiplo comum (mmc) dos inteiros dados. A quantidade de argumentos é variável.

- `log(x, base=math.e)`

Retorna o logaritmo de `x` na `base` dada. Caso não seja dada, a base usada é o número `e = 2.71828...`

- `sqrt(x)`

Retorna a raiz quadrada de `x`.

módulo `random`

- `choice(seq)`

Retorna uma elemento aleatório de `seq`, que deve ser uma sequência não vazia.

- `randrange(início, limite=None, passo=1)`

Retorna um elemento aleatório de `range(início, limite, passo)`. Os argumentos `limite` e `passo` são opcionais e têm a mesma interpretação que em `range`.

- `uniform(a, b)`

Retorna um `float` aleatório entre `a` e `b` (inclusive).