

MAC0110 Introdução à Computação

BACHARELADO EM ESTATÍSTICA, MATEMÁTICA E MATEMÁTICA APLICADA

Terceira Prova – 22 de junho de 2017

Nome: _____

Assinatura: _____

Nº USP: _____

Instruções:

1. Não destaque as folhas deste caderno. A prova pode ser feita a lápis.
2. A prova consta de 2 questões cada uma com 3 itens. Verifique antes de começar a prova se o seu caderno está completo.
3. As questões podem ser resolvidas em qualquer página. Ao escrever uma solução (ou parte dela) em página diferente do enunciado, escreva QUESTÃO X em letras ENORMES junto da solução.
4. As soluções devem ser em Python. Você pode usar apenas recursos de Python vistos em aula. Cuidado com a legibilidade e, principalmente, com a TABULAÇÃO.
5. As soluções não precisam verificar consistência de dados.
6. Não é permitido o uso de folhas avulsas para rascunho, a consulta a livros, apontamentos, colegas ou equipamentos eletrônicos. Desligue o seu celular e qualquer equipamento que possa perturbar o andamento da prova.

DURAÇÃO DA PROVA: 100 minutos

GOOD CODERS...



... KNOW WHAT THEY'RE DOING

Questão	Valor	Nota
1(a)	2,0	
1(b)	2,0	
1(c)	1,0	
2(a)	2,0	
2(b)	1,0	
2(c)	2,0	
Total	10,0	

Questão 1 (vale 5,0 pontos)

Essa questão possui 3 itens. Antes de resolver qualquer item, **leia toda a questão**, ou seja, o enunciado dos 3 itens.

Nessa questão você deve demonstrar o seu raciocínio computacional e sua habilidade em manipular strings e listas. Você **não deve utilizar** os métodos `split()` ou `strip()`. Caso você utilize um deles em sua solução, a sua questão não será corrigida e receberá nota ZERO.

Item 1(a) (vale 2,0 pontos)

Em computação um **caractere branco** é qualquer caractere ou série de caracteres que representam algum espaço vertical ou horizontal. Os caracteres ' ', '\n' e '\t' são exemplos de caracteres brancos que vocês conhecem.

Sem usar os métodos `split()` ou `strip()`, escreva uma função `separa_palavras()` como especificada a seguir.

```
BRANCO = ' \t\n'  
ESPACO = ' '
```

```
def separa_palavras( texto ):  
    ''' (str) -> list  
    Recebe um string `texto` e retorna uma lista contendo todas as palavras no texto.  
    Uma palavra é uma sequência de caracteres não brancos consecutivos.
```

Exemplos:

```
>>> separa_palavras(" Batatinha \n quando \t nasce esparrama a rama pelo chão.\n")  
['Batatinha', 'quando', 'nasce', 'esparrama', 'a', 'rama', 'pelo', 'chão.']  
>>> separa_palavras( " \n\t " )  
[]  
>>> separa_palavras( "um, \n 2 2,\ntrês.\n" )  
['um,', '2', '2,', 'três.']  
'''
```


Item 1(b) (vale 2,0 pontos)

Sem usar os métodos `split()` ou `strip()`, escreva um função `justifica_direita()` como especificada a seguir.

```
def justifica_direita( lp, nc ):
    ''' (list, int) -> str
```

Recebe uma lista `lp` com palavras e um inteiro `nc` indicando o número de colunas do texto formato com as palavras em `lp`. O string retornado deve representar o texto em `nc` colunas e alinhado à direita. A função pode supor que o comprimento de uma palavra é menor ou igual a `nc`.

Exemplos:

```
>>> lp=['0123456789','Aquarela','do','Brasil','em','verde','e','amarelo.']
>>> nc = 10
>>> justifica_direita (lp, nc)
'0123456789\n  Aquarela\n do Brasil\nem verde e\n  amarelo.'
>>> print(justifica_direita (lp, nc))
0123456789
  Aquarela
 do Brasil
em verde e
  amarelo.
>>> nc = 15
>>> lp[0] = "012345678901234"
>>> lp
['012345678901234', 'Aquarela', 'do', 'Brasil', 'em', 'verde', 'e', 'amarelo.']
>>> justifica_direita (lp, nc)
'012345678901234\n  Aquarela do\nBrasil em verde\n      e amarelo.'
>>> print(justifica_direita (lp, nc))
012345678901234
  Aquarela do
Brasil em verde
      e amarelo.
...
'''
```


Item 1(c) (vale 1,0 ponto)

Sem usar os métodos `split()` ou `strip()` e utilizando obrigatoriamente as funções `separa_palavras()` e `justifica_direita()`, escreva um programa (= `main()`) que leia um inteiro `nc` e um string `texto` e imprime o texto em `nc` colunas e alinhado à direita.

Você pode supor que o comprimento de uma palavra é sempre menor ou igual a `nc`.

Exemplos

Para `nc = 10` e o

`texto = '0123456789 Aquarela\n do\n\t Brasil em verde e \n amarelo.'`, o programa deve imprimir:

```
0123456789
  Aquarela
  do Brasil
  em verde
e amarelo.
```

Para `nc = 15` e o

`texto = '012345678901234 Aquarela do Brasil em verde e amarelo.'`, o programa deve imprimir:

```
012345678901234
  Aquarela do
Brasil em verde
  e amarelo.
```


Questão 2 (vale 5,0 pontos)

As constantes definidas abaixo podem ser utilizadas nessa questão, sem precisar redefini-las. Observe que há apenas 4 elementos possíveis em um mapa simplificado, e também apenas 4 movimentos possíveis.

```
# caracteres especiais em uma RLE
VAZIA      = '-'
NOVA_LINHA = '|'
DIGITOS    = '0123456789"

# elementos possíveis em um mapa
PAREDE     = '#'
PISO_VAZIO = ' '
MARCA_VAZIA = '.'
CAIXA_NO_PISO = '$'
CAIXA_NA_MARCA = '*'
JOGADOR_NO_PISO = '@'
JOGADOR_NA_MARCA = '+'

# movimentos
BAIXO     = 'b'
CIMA      = 'c'
DIR       = 'd'
ESQ      = 'e'
```

Item 2(a) (vale 2,0 pontos)

Escreva uma função `carregue_mapa()` como especificada a seguir.

```
def carregue_mapa():
    '''(None) -> matriz

    Função que lê um string com a descrição de um mapa de sokoban
    no formato RLE e cria e retorna uma matriz que representa o mapa.
    A matriz retornada é uma lista de listas de caracteres.
    '''
```

Por exemplo, para o string `'2-3#|-#-@-#|#2-$2-#|#4-$#|#2-$2-#|-#3-#|2-3#'` a função deve retornar a matriz

```
[[' ', ' ', '#', '#', '#'],
 [' ', '#', ' ', '@', ' ', '#'],
 ['#', ' ', ' ', '$', ' ', ' ', '#'],
 ['#', ' ', ' ', ' ', ' ', '$', '#'],
 ['#', ' ', ' ', '$', ' ', ' ', '#'],
 [' ', '#', ' ', ' ', ' ', '#'],
 [' ', ' ', '#', '#', '#']]
```


Item 2(b) (vale 1,0 ponto)

Escreva uma função `sokoban_resolvido()` como especificada a seguir.

```
def sokoban_resolvido( mapa ):  
    '''(matriz) -> bool  
    Recebe em `mapa` uma matriz que representa um sokoban. Se todas as caixas estão  
    sobre marcas a função retorna True, em caso contrário a função retorna False.  
    '''
```

Por exemplo, para

```
    0   1   2   3   4  
+---+---+---+---+---+  
0 |   | # | # | # |   |  
+---+---+---+---+---+  
1 | # | . | $ | @ | # |  
+---+---+---+---+---+  
2 |   | # | # | # |   |  
+---+---+---+---+---+  
                    mapa1
```

```
    0   1   2   3   4  
+---+---+---+---+---+  
0 |   | # | # | # |   |  
+---+---+---+---+---+  
1 | # | * | @ |   | # |  
+---+---+---+---+---+  
2 |   | # | # | # |   |  
+---+---+---+---+---+  
                    mapa2
```

temos que `sokoban_resolvido(mapa1)` retorna False e `sokoban_resolvido(mapa2)` retorna True.

No item a seguir você deverá utilizar, **sem escrevê-las**, as funções `mova_zelador()` e `to_string()`, especificadas a seguir.

```
def mova_zelador( mapa, mov ):
    '''(matriz, str) -> bool_1, bool_2

    Recebe em `mapa` uma matriz que representa um sokoban e em `mov` um movimento
    (= 'b', 'c', 'd' ou 'e') para o zelador.

    Se o mov é um movimento válido a função move o zelador, atualiza o mapa e
    retorna True como bool_1. Se uma caixa foi movida pelo zelador bool_2 é True,
    em caso contrário bool_2 é False.

    Se o movimento é inválido, a função retorna False, False.
    '''

def to_string( mapa ):
    '''(list) -> str

    Recebe em `mapa` uma matriz que representa um sokoban. A função cria e
    retorna um string que representa o mapa. Esse string é usado pela função
    print() para exibir o mapa.
    '''
```

Item 2(c) (vale 2,0 pontos)

Escreva um programa (= `main()`) que inicialmente

- carregue um mapa de sokoban e
- imprima o mapa carregado.

Em seguida o programa passa a

- ler um movimento do zelador (= 'b', 'c', 'd', 'e') ou 's' para *sair*;
- procurar realizar o movimento do zelador;
- imprimir uma mensagem indicando se o movimento é válido ou inválido; e
- imprimir o mapa atualizado.

O programa deve repetir os passos acima até que um 's' seja digitado ou o sokoban tenha sido resolvido. Se o sokoban tiver sido resolvido imprima **Parabéns!**, em caso contrário imprima **Deu ruim!**,

O seu programa deve utilizar, **obrigatoriamente**, todas as funções descritas anteriormente: `carregue_mapa()`, `sokoban_resolvido()`, `mova_zelador()` e `to_string()`. Você pode fazer esta questão mesmo que não tenha escrito a função `carregue_mapa()` ou a função `sokoban_resolvido()`.

Um exemplo de execução do programa está na próxima página O seu programa deve imprimir mensagens exatamente como nesse exemplo.

Configuração inicial:

```
  0  1  2  3  4  5  6
+---+---+---+---+---+---+
0 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
1 |  | # |  | @ |  | # |  |
+---+---+---+---+---+---+
2 | # |  |  | $ |  |  | # |
+---+---+---+---+---+---+
3 | # |  |  |  |  | $ | # |
+---+---+---+---+---+---+
4 | # |  |  | $ |  |  | # |
+---+---+---+---+---+---+
5 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
6 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
```

Digite o movimento do zelador: b

>>>> Movimento 'b' é válido
>>>> Zelador moveu uma caixa

```
  0  1  2  3  4  5  6
+---+---+---+---+---+---+
0 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
1 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
2 | # |  |  | @ |  | # |  |
+---+---+---+---+---+---+
3 | # |  |  | $ |  | $ | # |
+---+---+---+---+---+---+
4 | # |  |  | $ |  |  | # |
+---+---+---+---+---+---+
5 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
6 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
```

Digite o movimento do zelador: b

>>>> Movimento 'b' é inválido

Digite o movimento do zelador: d

```
>>>> Movimento 'd' é válido
  0  1  2  3  4  5  6
+---+---+---+---+---+---+
0 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
1 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
2 | # |  |  |  | @ |  | # |
+---+---+---+---+---+---+
3 | # |  |  | $ |  | $ | # |
+---+---+---+---+---+---+
4 | # |  |  | $ |  |  | # |
+---+---+---+---+---+---+
5 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
6 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
```

Digite o movimento do zelador: c

>>>> Movimento 'c' é válido

```
  0  1  2  3  4  5  6
+---+---+---+---+---+---+
0 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
1 |  | # |  |  | @ | # |  |
+---+---+---+---+---+---+
2 | # |  |  |  |  |  | # |
+---+---+---+---+---+---+
3 | # |  |  | $ |  | $ | # |
+---+---+---+---+---+---+
4 | # |  |  | $ |  |  | # |
+---+---+---+---+---+---+
5 |  | # |  |  |  | # |  |
+---+---+---+---+---+---+
6 |  |  | # | # | # |  |  |
+---+---+---+---+---+---+
```

Digite o movimento do zelador: c

>>>> Movimento 'c' é inválido

Digite o movimento do zelador: d

>>>> Movimento 'd' é inválido

Digite o movimento do zelador: s

Deu ruim!

>>>

