**Universidade de Brasília**

Departamento de Ciência da Computação

Introdução à Ciência da Computação - 113913

Gabarito da Lista de Exercícios 6

Listas

**Observações:**

* As listas de exercícios serão corrigidas por um **corretor automático,** portanto é necessário que as entradas e saídas do seu programa estejam conforme o padrão especificado em cada questão (exemplo de entrada e saída). Por exemplo, a não ser que seja requisitado na questão, não use mensagens escritas durante o desenvolvimento do seu código como “Informe a primeira entrada”. Estas mensagens não são tratadas pelo corretor, portanto a correção irá resultar em **resposta errada**, mesmo que seu código esteja correto;
* As instâncias de Entrada serão as usadas pelo corretor e suas saídas deve estar **iguais** às apresentadas em Instâncias de Saída.

Acesso Remoto

1 N = int(input())

2

3 products = []

**4 for** i **in** range(N):

5     products = [input()] + products

6

**7 print**(", ".join(products))

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 |  |
| 1  abacate | abacate |
| 10  when  unknown  printer  galley  type  scrambled  when  unknown  print  galley | galley, print, unknown, when, scrambled, type, galley, printer, unknown, when |
| 1  a b c d e f g h i j k l m n o p q r s t u v w | a b c d e f g h i j k l m n o p q r s t u v w |
| 5  roberto abraçou  um macaco dançante  isso não33 é nome de jogo  caracteres espe\*\*\*ciais!!!  que loucura, jovem!!!!!1!!! | que loucura, jovem!!!!!1!!!, caracteres espe\*\*\*ciais!!!, isso não33 é nome de jogo, um macaco dançante, roberto abraçou |
| 49  Fusce  malesuada  sem  justo,  id  posuere  mauris  facilisis  id.  Sed  tempus  nulla  sed  magna  cursus  interdum.  Nam  quis  consequat  est,  ac  pharetra  dui.  Aenean  nisl  tortor,  hendrerit  rutrum  rutrum  non,  semper  nec  lectus.  Aenean  egestas  vel  nisi  vel  varius.  Pellentesque  sagittis  rhoncus  sapien  eget  semper.  Curabitur  finibus  vitae  massa | massa, vitae, finibus, Curabitur, semper., eget, sapien, rhoncus, sagittis, Pellentesque, varius., vel, nisi, vel, egestas, Aenean, lectus., nec, semper, non,, rutrum, rutrum, hendrerit, tortor,, nisl, Aenean, dui., pharetra, ac, est,, consequat, quis, Nam, interdum., cursus, magna, sed, nulla, tempus, Sed, id., facilisis, mauris, posuere, id, justo,, sem, malesuada, Fusce |
| 4 | , , , |
| 3  \*\*\*  !!!  &&& | &&&, !!!, \*\*\* |
| 4  repeated  inputs  repeated  inputs | inputs, repeated, inputs, repeated |
| 7  rep3a ted  inpu ts  rep3a ted  wi7 h  sp3c1\*l  sym bols  ANDCAPS | ANDCAPS, sym bols, sp3c1\*l, wi7 h, rep3a ted, inpu ts, rep3a ted |

Bella e seus amigos

1 N = int(input())

2

3 convidados = []

**4 for** i **in** range(N):

5     convidados += [input()]

6

**7 print**("Cuidado!" **if** "André" **in** convidados **else** "Seguro!")

Na linha 7 nós temos o que é conhecido como “operador ternário”. O operador ternário verifica o valor de uma expressão dependendo da validade de um teste.

No caso, testamos **"André" in convidados** e, se verdadeiro, retornamos “Cuidado!”, caso contrário, “Seguro!”.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 | Seguro! |
| 1  André | Cuidado! |
| 1  Abacate | Seguro! |
| 2  RobErto  JurEma | Seguro! |
| 2  ANDRE  Andre | Seguro! |
| 4  paulo  joão  roberto  André | Cuidado! |
| 49  Fusce  malesuada  sem  justo,  id  posuere  mauris  facilisis  id.  Sed  tempus  nulla  sed  magna  cursus  interdum.  Nam  quis  consequat  est,  ac  pharetra  dui.  Aenean  nisl  tortor,  hendrerit  rutrum  rutrum  non,  semper  nec  lectus.  Aenean  egestas  vel  nisi  vel  varius.  Pellentesque  sagittis  rhoncus  sapien  eget  semper.  Curabitur  finibus  vitae  massa | Seguro! |
| 49  Fusce  malesuada  sem  justo,  id  posuere  mauris  facilisis  id.  Sed  tempus  nulla  sed  magna  cursus  interdum.  Nam  quis  consequat  est,  André  pharetra  dui.  Aenean  nisl  tortor,  hendrerit  rutrum  rutrum  non,  semper  nec  lectus.  Aenean  egestas  vel  nisi  vel  varius.  Pellentesque  sagittis  rhoncus  sapien  eget  semper.  Curabitur  finibus  vitae  massa | Cuidado! |
| 4  André  André  André  André | Cuidado! |
| 5  André  André  André  André  André | Cuidado! |

Cake Store

1 F, P = [int(x) **for** x **in** input().split()]

2

3 fatias = list(range(F))

4 fatias[P] = "\*"

5

**6 for** i **in** range(F):

7     N, E = input().split()

8     E = int(E)

9

10     **del** fatias[E]

11     **if** "\*" **not** **in** fatias:

12         **print**(N)

13         **break**

Caso você não tenha entendido nada da primeira linha, ela é a mesma coisa de:

1 F, P = input().split()

2 F, P = [int(F), int(P)]

Ou ainda:

1 F, P = input().split()

2 F = int(F)

3 P = int(P)

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 1 0  a 0 | a |
| 2 0  CAPSDEVEMSERMANTIDOS 1  wiNNer\_harOld 0 | wiNNer\_harOld |
| 5 3  Hello 2  Darkness 3  My 0  Old 1  Friend 0 | Old |
| 10 0  Doesnt 0  Matter 1  All 2  The 0  Inputs 1  On 3  The 1  End 0  Cuz 1  It\_might\_be\_the\_first 0 | Doesnt |
| 10 9  But 0  It 0  Also 0  Can 0  Be 0  The 0  Very 0  Last 0  One 0  Winner 0 | Winner |
| 10 6  The 0  Winner 0  Also 0  Can 0  Be 0  The 0  Loser 0  and\_\_ 0  that 0  is\_very\_weird 0 | Loser |
| 49 23  Fusce 41  malesuada 32  sem 46  justo 30  id 29  posuere 37  mauris 8  facilisis 13  id 10  Sed 15  tempus 16  nulla 28  sed 21  magna 25  cursus 34  interdu 33  Nam 2  quis 27  consequat 1  est 29  André 24  pharetra 1  dui 9  Aenean 3  nisl 0  tortor 10  hendrerit 20  rutrum 5  rutrum 7  non 8  semper 16  nec 15  lectus 7  Aenean 0  egestas 12  vel 10  nisi 0  vel 0  varius 0  Pellentesque 5  sagittis 4  rhoncus 1  sapien 4  eget 3  semper 0  Curabitur 3  finibus 1  vitae 1  massa 0 | finibus |
| 3 2  Rex 0  Weevil 1  Moto 0 | Weevil |
| 3 1  Yurick 0  Renato 0  Rafael 0 | Renato |
| 3 0  Pedro 2  Lemos 1  Martins 0 | Martins |

Déficit de Memória

1 N = int(input())

2 O = input().split()

*3 # copia a lista O, sem usar a mesma referência*

4 F = O[:]

5

**6 for** i **in** range(5):

7     [B, D, Q] = input().split()

8     Q = int(Q)

9     *# inverta o valor se vamos mover para a esquerda*

10     Q = -Q **if** D == "E" **else** Q

11

12     *# encontre o índice local de B na lista*

13     indexOfB = F.index(B)

14     newIndexOfB = indexOfB + Q

15     *# remova ela da posição atual*

16     **del** F[indexOfB]

17     *# insira na posição atualizada*

18     F[newIndexOfB:newIndexOfB] = [B]

19

20 C = 0

**21 for** i **in** range(N):

22     **if** F[i] != O[i]:

23         C += 1

24

**25 print**(C)

A linha 4 (F = O[:]) é a sacada da questão, por mais complexa que ela pareça na parte de manipulação de lista. Sem essa linha é impossível modificar a cópia da lista sem modificar a original, o que torna impossível realizar a comparação simples.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 3  A B C  A E 0  B E 0  C E 0  A E 0  B E 0 | 0 |
| 5  F G H I J  G D 2  G E 1  G E 1  G D 2  G D 1 | 4 |
| 1  A  A E 0  A E 0  A E 0  A E 0  A E 0 | 0 |
| 3  G H I  H D 1  I E 1  H D 1  I E 1  H D 1 | 2 |
| 5  A N D R E  E E 3  N E 0  R E 4  R D 0  A D 3 | 5 |
| 6  Y U R I C K  C E 3  Y D 4  R D 0  R D 0  R D 0 | 2 |
| 10  B C D E F G H I J K  C E 1  G D 2  K E 7  F D 2  H E 2 | 10 |
| 26  Q W E R T Y U I O P A S D F G H J K L Z X C V B N M  C E 15  B E 13  Q D 20  D D 4  V E 10 | 21 |
| 36  Q W E R T Y U I O P A S D F G H J K L Z X C V B N M 0 1 2 3 4 5 6 7 8 9  0 E 25  W D 4  Y D 19  J E 10  V E 7 | 22 |
| 2  H I  H E 1  H D 1  H E 1  H D 1  I E 1 | 2 |

Elastiman

1 N = int(input())

2

3 M = []

**4 for** i **in** range(N):

5     M.append(input().split())

6

**7 for** i **in** reversed(range(N)):

8     **for** j **in** range(N):

9         *# se pudermos deixar algo cair*

10         **if** M[i][j] == ".":

11             *# e tiver algo em cima*

12             **if** i-1 >= 0 **and** M[i-1][j] == "o":

13                 M[i][j] = "o"

14                 M[i-1][j] = "."

15

**16 for** i **in** range(N):

17     **print**(“ ”.join(M[i]), end="")

18     **print**()

O **print**() na linha 18 serve para imprimir uma nova linha, ao final da linha da matriz.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 2  o o  . . | . .  o o |
| 2  o o  x . | o .  x o |
| 3  o o o  o o o  . . . | . . .  o o o  o o o |
| 3  o . o  . o .  x x x | . . .  o o o  x x x |
| 1  o | o |
| 1  x | x |
| 5  . . . . .  . . . . .  . . . . .  . . . . .  . . . . . | . . . . .  . . . . .  . . . . .  . . . . .  . . . . . |
| 4  . o . .  . x . .  o . . .  . . o . | . o . .  . x . .  . . . .  o . o . |
| 3  x x x  x x x  x x x | x x x  x x x  x x x |
| 4  o o o o  o o o o  o o o o  o o o o | o o o o  o o o o  o o o o  o o o o |

Florêncio Pede Ajuda

**print**("".join(list(map(**lambda** w: w.capitalize(), input().split(“\_")))))

Caso você não tenho entendido bem o que o código acima faz, o código abaixo faz exatamente a mesma coisa, só que em várias linhas.

**1 def** capitalize(word):

2     **return** word.capitalize()

3

4 W = input()

5 words = W.split("\_")

6

7 capitalizedWords = map(capitalize, words)

**8 print**(“".join(capitalizedWords))

Tirar um tempo para entender por que o código de cima funciona é um ótimo exercício, e vale o investimento de tempo.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| oneword | Oneword |
| a\_b\_c\_d | ABCD |
| ab\_cd | AbCd |
| whats\_happening\_ | WhatsHappening |
| \_hello\_darkness | HelloDarkness |
| \_no\_spaces\_here\_ | NoSpacesHere |
| abcdefghijkl\_mnopqrstu\_vwxyz | AbcdefghijklMnopqrstuVwxyz |
| n\_o\_p\_l\_a\_c\_e\_l\_i\_k\_e\_h\_o\_m\_e | NOPLACELIKEHOME |
| hello\_stranger | HelloStranger |
| \_\_\_\_\_\_\_wat\_\_\_\_\_\_iss\_this\_even\_\_possible | WatIssThisEvenPossible |

Hanoi

1 [H, P] = [int(x) **for** x **in** input().split()]

2

3 E = list(range(H))

4 E.reverse()

5

6 C = []

7 D = []

8

**9 def** move(n, source, target, idle, steps):

10     **if** n > 0:

11         steps = move(n-1, source, idle, target, steps)

12         **if** steps <= 0: **return** steps

13

14         steps -= 1

15         target.append(source.pop())

16         **if** steps <= 0: **return** steps

17

18         steps = move(n-1, idle, target, source, steps)

19         **if** steps <= 0: **return** steps

20

21     **return** steps

22

23 move(H, E, D, C, P)

24

**25 print**("%d %d %d" % (len(E), len(C), len(D)))

Tenha em mente que outras implementações dessa mesma solução são perfeitamente possíveis, inclusive sem utilizar recursividade.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 128 5 | 126 1 1 |
| 128 2 | 126 1 1 |
| 5 10 | 2 2 1 |
| 5 25 | 2 0 3 |
| 5 50 | 0 0 5 |
| 32 8 | 28 3 1 |
| 32 100 | 27 3 2 |
| 32 500 | 24 7 1 |
| 32 1000 | 23 1 8 |
| 32 1000000 | 15 3 14 |