**Universidade de Brasília**

Departamento de Ciência da Computação

Introdução à Ciência da Computação - 113913

Gabarito da Lista de Exercícios 7

Tuplas

**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.

A

1 perms = (   **lambda** a,b,c: (c,b,a),

2             **lambda** a,b,c: (a,c,b),

3             **lambda** a,b,c: (b,a,c)

4         )

5

6 org = ('A', 'B', 'C')

7

8 N = int(input())

9

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

11     p = int(input())

12     org = perms[p-1](\*org)

13

**14 print**(" ".join(org))

Caso você não saiba para que serve o \* na linha 12, ele é o operador de spread do python. Ele abre a lista ou a tupla e usa seus elementos como parâmetros da função, ou seja:

f(\*x) é a mesma coisa de f(x[0], x[1], x[2], …)

E caso você não entenda o que é um **lambda,** tenha em mente que:

1 **lambda** a,b,c: (c,b,a)

é a mesma coisa de

1 **def** fun(a,b,c):

2 **return** (c,b,a)

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 | A B C |
| 1  1 | C B A |
| 1  2 | A C B |
| 1  3 | B A C |
| 5  1  2  3  2  1 | C B A |
| 25  2  2  1  3  2  3  1  2  3  3  2  2  2  2  2  1  2  3  2  3  2  2  3  1  3 | B A C |
| 6  1  3  3  2  3  1 | B C A |
| 5  3  2  1  2  1 | C B A |
| 7  1  2  1  2  3  1  2 | A C B |
| 50  1  3  3  2  1  2  2  3  1  1  1  2  1  3  1  3  1  1  1  1  1  3  1  1  3  3  3  2  1  2  2  1  3  2  2  3  2  2  3  3  1  3  3  1  3  1  1  3  2  1 | B C A |

Brasil

*1 #coding: utf-8*

2

3 N = int(input())

4

5 servers = []

6

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

8     S, R = input().split()

9     R = int(R)

10

11     **if** R > 80:

12         servers.append((S, R))

13

14 o\_list = sorted(servers, key=**lambda** x: -x[1])

15

**16 for** server **in** o\_list:

17     **print**(server[0])

18

**19 if** len(o\_list) == 0:

20     **print**("Ninguém!")

A primeira linha garante que o acento agudo do print de fallback estará lá. Atente a ele, é possível que muitas respostas deem WA por conta desse pequeno detalhe.

Caso você esteja curioso e queira saber mais sobre o trecho de código mágico, entre em contato com o monitor mais próximo que eu tenho certeza que ele vai ter muito prazer em explicar em detalhes o que significa.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 | Ninguém! |
| 1  a 80 | Ninguém! |
| 10  Chan 66  Alexei 49  Leui 85  Arnie 2  Andre 43  Baugulf 33  Apostol 6  Roberto 92  Modred 4  Nikodemos 65 | Roberto  Leui |
| 26  a 0  b 2  c 4  d 8  e 16  f 32  g 64  h 100  i 95  j 93  k 85  l 1  m 3  n 9  o 27  p 81  q 99  r 90  s 80  t 70  u 5  v 6  w 17  x 65  y 98  z 91 | h  q  y  i  j  z  r  k  p |
| 2  a 81  b 80 | a |
| 5  a 81  b 82  c 83  d 84  e 85 | e  d  c  b  a |
| 7  a 81  b 80  a 97  c 30  a 0  t 10  e 26 | a  a |
| 3  a 0  b 1  c 2 | Ninguém! |
| 10  a 0  b 1  c 2  d 3  e 4  f 5  g 6  h 7  i 8  j 9 | Ninguém! |
| 1  a 100 | a |

Caviar

1 Xe, Ye = [int(x) **for** x **in** input().split()]

2 Xd, Yd = [int(x) **for** x **in** input().split()]

3 Xo, Yo = [int(x) **for** x **in** input().split()]

4

**5 if** Xe <= Xo <= Xd **and** Ye <= Yo <= Yd:

6     **print**("Cuidado!")

**7 else**:

8     **print**("Seguro!")

Novamente, o trecho

Xe, Ye = **[int(x) for x in input().split()]**

é equivalente a

xy = input().split()

Xe, Ye = int(xy[0]), int(xy[1])

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 1 1  100000000 100000000  0 0 | Seguro! |
| -10 -10  10 10  0 0 | Cuidado! |
| -20 10  10 30  50 15 | Seguro! |
| 10 10  15 15  15 15 | Cuidado! |
| -10 -20  10 20  11 21 | Seguro! |
| -30 -50  -20 -10  -30 -10 | Cuidado! |
| 10 10  10 10  11 11 | Seguro! |
| 10 10  10 10  10 10 | Cuidado! |
| -9999 -9999  9999 9999  10000 0 | Seguro! |
| 0 0  0 0  0 0 | Cuidado! |

Deus Ex Machina

1 N = int(input())

2

3 list\_of\_problems = []

4

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

6     P, S, D = input().split()

7     D = int(D)

8

9     list\_of\_problems.append((S, D))

10

*11 # from question B*

12 o\_list = sorted(list\_of\_problems, key=**lambda** x: -x[1])

13

**14 for** solution **in** o\_list:

15     **print**(solution[0], end="")

**16 print**() *# for newline*

Caso você não tenha ideia de onde veio o código da linha 12, verifique o enunciado da questão Brasil.

A função sorted é garantida de fazer uma ordenação estável, então a ordem de input é preservada no caso de empate.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 |  |
| 1  a a 1 | a |
| 3  a a 1  b b 2  c c 2 | bca |
| 3  a a 10  b b 5  c c 1 | abc |
| 3  a a 1  b b 5  c c 10 | cba |
| 5  uwehdwde d 1  iwudnieuw c 2  ndiuewwd e 3  a f 4  b g 5 | gfecd |
| 5  d uwei 10  c udwiendneiw 9  b yewedee 8  a iwemdwe 9  \_ \_ 7 | uweiudwiendneiwiwemdweyewedee\_ |
| 10  a a 1  a a 1  a a 1  a a 1  a a 1  a a 1  a a 1  a a 1  a a 1  a a 1 | aaaaaaaaaa |
| 8  duewn dewyhew 8  dweiu uiwejdw 7  dwnhehndew diewwd 8  dwiej ndwueindmw 10  dwjhenw dwnedwmd 6  dneiwdmew uidwjdeiw 1  dwmedow dwemodmw 1  dmwiedmeiw due 4 | ndwueindmwdewyhewdiewwduiwejdwdwnedwmddueuidwjdeiwdwemodmw |
| 49  Duis sit 5  amet odio 8  vitae diam 2  tempus aliquet. 3  Aenean fringilla 4  interdum risus, 4  id ultrices 2  erat rhoncus 8  ac. Sed 2  eu tellus 9  vestibulum leo 7  mattis aliquet 1  sit amet 6  vel leo. 3  Fusce eget 4  euismod enim. 8  Maecenas consectetur 3  vehicula diam, 1  ut interdum 6  justo commodo 3  id. Maecenas 5  iaculis lectus 7  metus. Fusce 6  porta, nisl 2  eget accumsan 5  rutrum, sem 2  nisi maximus 1  tellus, a 1  semper sapien 5  augue ut 5  augue. Vivamus 1  scelerisque, massa 0  et convallis 3  ullamcorper, mauris 9  massa condimentum 2  magna, nec 6  dictum augue 0  tellus in 0  elit. Quisque 1  in elit 2  sed neque 8  fermentum elementum. 9  Aliquam enim 7  diam, scelerisque 7  non dictum 9  a, molestie 1  nec ante. 3  Vestibulum ante 2  ipsum primis 5 | tellusmauriselementum.dictumodiorhoncusenim.nequeleolectusenimscelerisqueametinterdumFuscenecsitMaecenasaccumsansapienutprimisfringillarisus,egetaliquet.leo.consecteturcommodoconvallisante.diamultricesSednislsemcondimentumelitantealiquetdiam,maximusaVivamusQuisquemolestiemassaauguein |

Estrada

1 N = int(input())

2

**3 def** move(pos, D, Q):

4     **if** D == "N":

5         **return** (pos[0]+Q, pos[1])

6     **elif** D == "S":

7         **return** (pos[0]-Q, pos[1])

8     **elif** D == "L":

9         **return** (pos[0], pos[1]+Q)

10     **elif** D == "O":

11         **return** (pos[0], pos[1]-Q)

12

**13 def** format(y, x):

14     **return** (

15         -y **if** y < 0 **else** 0,

16         y **if** y > 0 **else** 0,

17         -x **if** x < 0 **else** 0,

18         x **if** x > 0 **else** 0

19     )

20

21 pos = (0, 0)

22

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

24     D, Q = input().split()

25     pos = move(pos, D, int(Q))

26

**27 print**("%d %d %d %d" % format(\*pos))

Evidentemente esta solução não precisa das funções move e format, mas achei um exemplo apropriado de funções que retornam mais de um valor.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0 | 0 0 0 0 |
| 4  N 1  S 1  O 1  L 1 | 0 0 0 0 |
| 10  N 7  L 28  S 14  O 10  N 19  N 10  N 28  S 100  S 45  O 10 | 95 0 0 8 |
| 5  O 5  O 5  O 5  O 5  O 5 | 0 0 25 0 |
| 5  L 10  L 10  L 10  L 10  L 10 | 0 0 0 50 |
| 6  S 3  S 3  S 3  S 3  S 3  S 3 | 18 0 0 0 |
| 20  S 22  S 63  L 27  O 83  O 10  O 9  S 100  O 32  S 76  N 11  S 56  L 68  L 55  N 42  S 61  O 21  O 20  L 31  S 93  N 69 | 349 0 0 6 |
| 26  N 283  N 201  S 511  N 914  S 443  O 700  S 513  S 692  N 92  S 329  L 187  O 69  S 657  N 542  S 706  N 645  S 2  S 946  O 641  S 77  L 398  O 242  O 315  N 852  L 964  N 371 | 976 0 418 0 |
| 47  O 628  S 782  S 782  L 433  S 987  O 810  O 72  O 34  N 759  L 337  S 883  O 596  L 74  S 299  S 890  L 954  N 11  L 183  O 392  L 550  N 93  S 969  S 697  N 721  S 88  L 95  N 308  N 433  O 742  S 467  O 913  N 414  N 161  N 394  N 213  S 133  O 779  S 954  N 755  S 500  S 417  S 480  L 197  N 876  S 764  O 463  O 108 | 4954 0 2714 0 |
| 19  O 86  S 64  L 102  O 335  N 250  O 799  N 163  N 19  O 766  N 594  O 775  L 210  O 487  O 925  L 3767  L 135  N 832  N 148  S 1904 | 0 38 0 41 |

Filosofia

1 N = int(input())

2

3 files = []

4

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

6     temp = input().split()

7     files.append((temp[0], temp[1:]))

8

9 tags = input().split()

10

11 request = list(filter(**lambda** f: list(filter(**lambda** tag: tag **in** tags, f[1])), files))

12

**13 for** r **in** request:

14     **print**(r[0])

A linha 11 pode parecer super complexa, e ela meio que é. Tente quebrá-la em passos para entender melhor o que está acontecendo.

**lambda x: y** cria uma função que toma x de argumento e retorna y, na hora.

list(filter(x, y)) cria uma lista a partir de y contendo somente os elementos que, quando alimentados para a função x, retornem true.

Consulte o monitor mais próximos para maiores dúvidas.

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 0  potato |  |
| 1  a b c d e  c | a |
| 2  a b c d e  e d c b a  a | d |
| 2  a b c d e  e d c b a  d | a |
| 2  a b c d e  e d c b a  c | a  d |
| 2  a b c d e  e d c b a  d a | a  d |
| 2  a b c d e  e a b c d  a |  |
| 2  a b c d e  e a b c d  d | a  a |
| 2  a b e h i  a e f g j  e f | a  a |
| 40  Sed at volutpat en im.  Sed non sollicitudin di am.  Vivamus turpis turpis, fa cilisis  eu gravida e u, volutpat  et nibh. Aenean fauc ibus  turpis in met us volutpat,  a malesuada libe ro pellentesque.  Morbi dictum sap ien ut  finibus consequat. Nu lla eget  rutrum tellus, ac lao reet  magna. Pellentesque a lo rem  quam. Integer rutrum scele risque  eros vitae ornare. a t.  Mauris ullamcorper frin gilla diam.  Cras non nisl con gue,  finibus ex in, pul vinar  ipsum. Vestibulum lec tus est,  pellentesque sit am et varius  in faucibus orci lu ctus  et ultrices posu ere cubilia  Curae; Fusce sceler isque ipsum  quis tempus sollic itudin. Pellentesque  eget odio com modo, lobortis  tellus sed, vesti bulum orci.  Pellentesque pu rus quam, tincidunt  ut varius quis, max imus  at tortor. Pellentesque e t  egestas risus, sit am et  ullamcorper ri sus. Fusce tincidunt  arcu eget eros. Pra esent  id diam male suada, eleifend  odio quis, cons equat nisi.  Aenean port titor enim id  justo congue orna re. Fusce  auctor ipsum lib ero, lobortis  molestie metus fau cibus in.  Proin ut sod ales elit,  sit amet acc umsan est.  Nullam condim entum quam eget  purus tristique aliq uet. Vivamus  volutpat tristique Pellentesque | Sed  eu  magna.  quis  at  purus |

Gabirint

1 N = int(input())

2

3 plays = []

4

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

6     J, I = input().split()

7     I = int(I)

8

9     plays.append((J, I))

10

11 current = plays[N-1][1]

**12 while** plays[current-1][1] != N:

13     current = plays[current-1][1]

14

**15 print**(plays[current-1][0])

| Instâncias de Entrada | Instâncias de Saída |
| --- | --- |
| 1  semGraca 1 | semGraca |
| 3  yurick 3  qualquer 3  yurick 1 | yurick |
| 5  yurick 3  rafael 5  remy 4  joao 2  manoel 5 | manoel |
| 10  abacate 8  rafael 3  potato 4  byebye 7  fägel 10  brutamontes 2  joelho 18  abacate 5  rafael 3  potato 1 | fägel |
| 10  verifique 10  se 10  ele 10  so 10  esta 10  checando 10  pela 10  presenca 10  de 10  N 6 | checando |
| 10  verifique 318  se 420  ele 9999  da 9999999  suporte 0  a -24  numeros 10  invalidos 7  :( 8  :) 9 | numeros |
| 5  🗣 5  💆 3  👳 1  💂 1  😬 3 | 🗣 |
| 5  5 5  3 3  1 1  1 1  3 3 | 5 |
| 3  >< 3  >< 1  >< 1 | >< |
| 6  a 5  b 4  c 4  d 1  e 10  f 6 | f |