```python
#!/usr/bin/python -tt
# Exercícios by Nick Parlante (CodingBat)

# A. dormir
# dia_semana é True para dias na semana
# feriado é True nos feriados
# você pode ficar dormindo quando é feriado ou não é
# dia semana
# retorne True ou False conforme você vá dormir ou não
def dormir(dia_semana, feriado):
  return not dia_semana or feriado

# B. alunos_problema
# temos dois alunos a e b
# a_sorri e b_sorri indicam se a e b sorriem
# temos problemas quando ambos estão sorrindo ou ambos
# não estão sorrindo
# retorne True quando houver problemas
def alunos_problema(a_sorri, b_sorri):
  return a_sorri == b_sorri

# C. soma_dobro
# dados dois números inteiros retorna sua soma
# porém se os números forem iguais retorna o dobro da soma
# soma_dobro(1, 2) -> 3
# soma_dobro(2, 2) -> 8
def soma_dobro(a, b):
  return 2*(a + b) if a == b else a + b

# D. diff21
# dado um inteiro n retorna a diferença absoluta entre n e 21
# porém se o número for maior que 21 retorna dobro da diferença absoluta
# diff21(19) -> 2
# diff21(25) -> 8
# dica: abs(x) retorna o valor absoluto de x
def diff21(n):
  return 2*abs(n - 21) if n > 21 else abs(n - 21)
```

```python
# H. Anagrama
# Verifique se duas palavras são anagramas,
# isto é são uma é permutação das letras da outra
# anagrama('aberto', 'rebato') = True
# anagrama('amor', 'ramo') = True
# anagrama('aba', 'baba') = False
def anagrama(s1, s2):
  return sorted(s1) == sorted(s2)

def test(obtido, esperado):
  if obtido == esperado:
    prefixo = ' Parabéns!'
  else:
    prefixo = ' Ainda não'
  print ('%s obtido: %s esperado: %s' % (prefixo, repr(obtido), repr(esperado)))
```

```python
# D. Dada uma lista de números retorna uma lista sem os elementos repetidos
def remove_iguais(nums):
  return list(set(nums))

# E. Cripto desafio!!
# Dada uma frase, você deve retirar todas as letras repetidas das palavras
# e ordenar as letras que sobraram
# Exemplo: 'ana e mariana gostam de banana' vira 'an e aimnr agmost de abn'
# Dicas: tente transformar cada palavra em um conjunto, depois tente sortear
# as letras e montar uma string com o resultado.
# Utilize listas auxiliares se facilitar
def cripto(frase):
  return ' '.join([''.join(
            sorted(set(p)))
            for p in frase.split()])

# F. Derivada de um polinômio
# Os coeficientes de um polinômio estão numa lista na ordem do seu grau.
# Você deverá devolver uma lista com os coeficientes da derivada.
# Exemplo: [3, 2, 5, 2] retorna [2, 10, 6]
# A derivada de 3 + 2x + 5x^2 + 2x^3 é 2 + 10x + 6x^2
def derivada(coef):
  return [c*grau for c,grau in enumerate (coef)][1:]

# G. Soma em listas invertidas
# Colocamos os dígitos de dois números em listas ao contrário
# 513 vira [3, 1, 5] e 295 vira [5, 9, 2]
# [3, 1, 5] + [5, 9, 2] = [8, 0, 8]
# pode supor que n1 e n2 tem o mesmo número de dígitos
# Não vale converter a lista em número para somar diretamente
def soma(n1, n2):
  r = []
  v1 = 0
  for x, y in zip(n1, n2):
    n = (x + y) % 10 + v1
    v1 = (x + y) // 10
    r.append(n)
  if v1 != 0: r.append(v1)
  return r
# C. sort_last
# Dada uma lista de tuplas não vazias retorna uma tupla ordenada
# por ordem crescente do último elemento
# Exemplo [(1, 7), (1, 3), (3, 4, 5), (2, 2)] retorna
# [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
# Dica: use key=função que você definiu e que retorna o último elemento
def sort_last(tuples):
  return sorted(tuples, key=last)

```

```python
# A. fim_igual
# Dada uma lista de strings, retorna o número de strings
# com tamanho >= 2 onde o primeiro e o último caracteres são iguais
def fim_igual(words):
  return len([w for w in words
         if len(w) >= 2 and w[0] == w[-1]])
  k = 0
  for word in words:
    if len(word) >= 2 and word[0] == word[-1]:
      k = k + 1
  return k

# B. x_antes
# Dada uma lista de strings retorna uma lista onde todos os elementos
# que começam com x ficam sorteados antes
# Exemplo ['mix', 'xyz', 'apple', 'xanadu', 'aardvark'] retorna
# ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
# Dica: monte duas listas separadas e junte-as no final
def x_antes(words):
  x = []
  outros = []
  for w in words:
    if w.startswith('x'): #w[0] == 'x'
      x.append(w)
    else:
      outros.append(w)
  return sorted(x) + sorted(outros)

# LAB(begin solution)
# Extract the last element from a tuple -- used for custom sorting below.
def last(a):  return a[-1]
# LAB(end solution)
```

```python
# J. zeros finais
# Verifique quantos zeros há no final de um número inteiro positivo
# Exemplo: 10010 tem 1 zero no fim e 908007000 possui três
def zf(n):
  n = str(n)[::-1]
  k = 0
  while n[k] == '0':
    k = k + 1
  return k
  a = int(str(n)[::-1])
  return len(str(n)) - len(str(a))

# K. conta 2
# Verifique quantas vezes o dígito 2 aparece entre 0 e n-1
# Exemplo: para n = 20 o dígito 2 aparece duas vezes entre 0 e 19
def conta2(n):
  s = ''
  for i in range(n):
    s = s + str(i)
  return s.count('2')

# L. inicio em potencia de 2
# Dado um número inteiro positivo n retorne a primeira potência de 2
# que tenha o início igual a n
# Exemplo: para n = 65 retornará 16 pois 2**16 = 65536
def inip2(n):
  k = 0
  while True:
    pot = str(2**k)
    if pot.startswith(str(n)):
      return k
    k = k + 1
```

```python
# G. verbing
# Dada uma string, caso seu comprimento seja pelo menos 3,
# adiciona 'ing' no final
# Caso a string já termine em 'ing', acrescentará 'ly'.
def verbing(s):
  if len(s) >= 3:
    if s[-3:] != 'ing':
      s = s + 'ing'
    else:
      s = s + 'ly'
  return s

# H. not_bad
# Dada uma string, procura a primeira ocorrência de 'not' e 'bad'
# Se 'bad' aparece depois de 'not' troca 'not' ... 'bad' por 'good'
# Assim 'This dinner is not that bad!' retorna 'This dinner is good!'
def not_bad(s):
  n = s.find('not')
  b = s.find('bad')
  if b > n:
    s = s[:n] + 'good' + s[b+3:]
  return s

# I. inicio_final
# Divida cada string em dois pedaços.
# Se a string tiver um número ímpar de caracteres o primeiro pedaço terá um caracter a mais,
# Exemplo: 'abcde', divide-se em 'abc' e 'de'.
# Dadas 2 strings, a e b, retorna a string
#  a_inicio + b_inicio + a_final + b_final
def inicio_final(a, b):
  a_meio = len(a) // 2
  b_meio = len(b) // 2
  if len(a) % 2 == 1:  # aumenta 1 se len é impar
    a_meio = a_meio + 1
  if len(b) % 2 == 1:
    b_meio = b_meio + 1
  return a[:a_meio] + b[:b_meio] + a[a_meio:] + b[b_meio:]

# F. busca (COMP 89 IME-USP)
# Verifique quantas ocorrências de uma palavra há numa frase
# frase = 'ana e mariana gostam de banana'
# palavra = 'ana'
# busca ('ana e mariana gostam de banana', 'ana') == 4
# Hall of Fame Victor H. Panisa, 1a turma Python para Zumbis
def busca(frase, palavra):
  return len([k for k in range(len(frase))
      if frase[k:k+len(palavra)] == palavra])

```

```python
# A. donuts
# Para um inteiro n retorna uma string na forma 'Número de donuts: <n>'
# onde n é o valor passado como argumento.
# Caso n >= 10 devo retornar 'muitos' em lugar do número.
# donuts(5) returns 'Número de donuts: 5'
# donuts(23) returns 'Número de donuts: muitos'
def donuts(n):
  return f'Número de donuts: {"muitos" if n >= 10 else n}'

# B. pontas
# Dada uma string s, retorna uma string com as duas primeiras e as duas
# últimas letras da string original s
# Assim 'palmeiras' retorna 'paas'
# No entanto, se a string tiver menos que 2 letras, retorna uma string vazia
def pontas(s):
  return '' if len(s) < 2 else s[:2] + s[-2:]

# C. fixa_primeiro
# Dada uma string s, retorna uma string onde todas as ocorrências
# do primeiro caracter são trocados por '*', exceto para o primeiro
# Assim 'abacate' retorna 'ab*c*te'
# Dica: use s.replace(stra, strb)
def fixa_primeiro(s):
  return s[0] + s[1:].replace(s[0], '*')

# D. mistura2
# Sejam duas strings a e b
# Retorno uma string '<a> <b>' separada por um espaço
# com as duas letras trocadas de cada string
#   'mix', pod' -> 'pox mid'
#   'dog', 'dinner' -> 'dig donner'
def mistura2(a, b):
  return b[:2] + a[2:] + ' ' + a[:2] + b[2:]

# E. palindrome
# Verifique se uma string é palíndrome
#   palindrome('asa') True
#   palindrome('casa') False
def palindrome(s):
  return s == s[::-1]
```

```python
# L. soma_na_lista
# Verifica se n é soma de dois números distintos da lista
# soma_na_lista(5, [1, 2, 3, 4]) -> True
# soma_na_lista(9, [1, 2, 3, 4]) -> False
# soma_na_lista(0, [1, 2, 3, 4]) -> False
# soma_na_lista(8, [1, 2, 3, 4]) -> False
# soma_na_lista(4, [2, 2, 2, 2]) -> False
# soma_na_lista(4, [2, 2, 1, 3]) -> True
# Hall of Fame: Raphael Castro 1a Turma Python para Zumbis
def soma_na_lista(n, lista):
  return n in [x + y for x in lista for y in lista if x != y]

# M. desafio! faça somente se já tiver acabado o EP1 e todas as listas
# Fila de tijolos sem usar loops
# queremos montar uma fila de tijolos de um tamanho denominado meta
# temos tijolos pequenos (tamanho 1) e tijolos grandes (tamanho 5)
# retorna True se for possível montar a fila de tijolos
# é possível uma solução sem usar for ou while
# fila_tijolos(3, 1, 8) -> True
# fila_tijolos(3, 1, 9) -> False
# fila_tijolos(3, 2, 10) -> True
def fila_tijolos(n_peq, n_gra, meta):
  return n_peq >= meta % 5 and n_peq + 5 * n_gra >= meta

# H. end_other
# as duas strings devem ser convertidas para minúsculo via lower()
# depois disso verifique que no final da string b ocorre a string a
# ou se no final da string a ocorre a string b
# end_other('Hiabc', 'abc') -> True
# end_other('AbC', 'HiaBc') -> True
# end_other('abc', 'abXabc') -> True
def end_other(a, b):
  a = a.lower()
  b = b.lower()
  return a.endswith(b) or b.endswith(a)

# I. count_evens
# conta os números pares da lista
# count_evens([2, 1, 2, 3, 4]) -> 3
# count_evens([2, 2, 0]) -> 3
# count_evens([1, 3, 5]) -> 0
def count_evens(nums):
  return len([x for x in nums if x % 2 == 0])

# J. sum13
# retorna a soma dos números de uma lista
# 13 dá azar, você deverá ignorar o 13 e todos os números à sua direita
# sum13([1, 2, 2, 1]) -> 6
# sum13([1, 1]) -> 2
# sum13([1, 2, 2, 1, 13]) -> 6
# sum13([13, 1, 2, 3, 4]) -> 0
def sum13(nums):
  if 13 in nums:
    return sum(nums[:nums.index(13)])
  return sum(nums)

# K. has22
# Verifica se na lista de números inteiros aparecem dois 2 consecutivos
# has22([1, 2, 2]) -> True
# has22([1, 2, 1, 2]) -> False
# has22([2, 1, 2]) -> False
def has22(nums):
  return '2, 2' in str(nums)

```

```python
# D. double_char
# retorna os caracteres da string original duplicados
# double_char('The') -> 'TThhee'
# double_char('AAbb') -> 'AAAAbbbb'
# double_char('Hi-There') -> 'HHii--TThheerree'
def double_char(s):
  return ''.join([c+c for c in s])

# E. count_hi
# conta o número de vezes que aparece a string 'hi'
# count_hi('abc hi ho') -> 1
# count_hi('ABChi hi') -> 2
# count_hi('hihi') -> 2
def count_hi(s):
  return s.count('hi')

# F. cat_dog
# verifica se o aparece o mesmo número de vezes 'cat' e 'dog'
# cat_dog('catdog') -> True
# cat_dog('catcat') -> False
# cat_dog('1cat1cadodog') -> True
def cat_dog(s):
  return s.count('cat') == s.count('dog')

# G. count_code
# conta quantas vezes aparece 'code'
# a letra 'd' pode ser trocada por outra qualquer
# assim 'coxe' ou 'coze' também são contadas como 'code'
# count_code('aaacodebbb') -> 1
# count_code('codexxcode') -> 2
# count_code('cozexxcope') -> 2
def count_code(s):
  cont = 0
  for k in range(len(s)-3):
    if s[k:k+2] == 'co' and s[k+3] == 'e':
      cont += 1
  return cont

```

```python
# A. near_ten
# Seja um n não negativo, retorna True se o número está a distância de
# pelo menos dois de um múltiplo de dez. Use a função resto da divisão.
# near_ten(12) -> True
# near_ten(17) -> False
# near_ten(19) -> True
def near_ten(n):
  return n % 10 <= 2 or n % 10 >= 8

# B. lone_sum
# Soma maluca: some os números inteiros a, b, e c
# Se algum número aparecer repetido ele não conta na soma
# lone_sum(1, 2, 3) -> 6
# lone_sum(3, 2, 3) -> 2
# lone_sum(3, 3, 3) -> 0
def lone_sum(a, b, c):
  if a == b == c:
    return 0
  if a == b:
    return c
  if b == c:
    return a
  if a == c:
    return b
  return a + b + c

# C. luck_sum
# Soma três inteiros a, b, c
# Se aparecer um 13 ele não conta e todos os da sua direita também
# lucky_sum(1, 2, 3) -> 6
# lucky_sum(1, 2, 13) -> 3
# lucky_sum(1, 13, 3) -> 1
def lucky_sum(a, b, c):
  if a == 13:
    return 0
  if b == 13:
    return a
  if c == 13:
    return a + b
  return a + b + c
```

```python
# J. alarm_clock
# day: 0=domingo, 1=segunda, 2=terça, ..., 6=sábado
# vacation = True caso você esteja de férias
# o retorno é uma string que diz quando o despertador tocará
# dias da semana '07:00'
# finais de semana '10:00'
# a menos que você esteja de férias, neste caso:
# dias da semana '10:00'
# finais de semana 'off'
# alarm_clock(1, False) -> '7:00'
# alarm_clock(5, False) -> '7:00'
# alarm_clock(0, False) -> '10:00'
def alarm_clock(day, vacation):
  if vacation:
    return 'off' if day in (0, 6) else '10:00'
  else:
    return '10:00' if day in (0, 6) else '7:00'
```

```python
# H. squirrel_play
# os esquilos na FATEC brincam quando a temperatura está entre 60 e 90
# graus Fahreneit (são estrangeiros e o termômetro é diferente rs)
# caso seja verão, então a temperatura superior é 100 no lugar de 90
# retorne True caso os esquilos brinquem
# squirrel_play(70, False) -> True
# squirrel_play(95, False) -> False
# squirrel_play(95, True) -> True
def squirrel_play(temp, is_summer):
  return 60 <= temp <= 100 if is_summer else 60 <= temp <= 90

# I. pego_correndo
# você foi pego correndo
# o resultado será:
# sem multa = 0
# multa média = 1
# multa grave = 2
# velocide <= 60 sem multa
# velocidade entre 61 e 80 multa média
# velocidade maior que 81 multa grave (cidade do interior)
# caso seja seu aniversário a velocidade pode ser 5 km/h maior em todos os casos
# pego_correndo(60, False) -> 0
# pego_correndo(65, False) -> 1
# pego_correndo(65, True) -> 0
def pego_correndo(speed, is_birthday):
  if is_birthday:
    speed = speed - 5
  if speed <= 60:
    return 0
  if 61 <= speed <= 80:
    return 1
  return 2
```

```python
# E. sum2
# Dada uma lista de inteiros de qualquer tamanho
# retorna a soma dos dois primeiros elementos
# se a lista tiver menos de dois elementos, soma o que for possível
def sum2(nums):
  return sum(nums[:2])

# F. middle_way
# sejam duas listas de inteiros a e b
# retorna uma lista de tamanho 2 contendo os elementos do
# meio de a e b, suponha que as listas tem tamanho ímpar
# middle_way([1, 2, 3], [4, 5, 6]) -> [2, 5]
# middle_way([7, 7, 7], [3, 8, 0]) -> [7, 8]
# middle_way([5, 2, 9], [1, 4, 5]) -> [2, 4]
def middle_way(a, b):
  return [a[len(a)//2], b[len(b)//2]]

# G. date_fashion
# você e sua namorada(o) vão a um restaurante
# eu e par são as notas das suas roupas de 0 a 10
# quanto maior a nota mais chique vocês estão vestidos
# o resultado é se vocês conseguiram uma mesa no restaurante:
# 0=não 1=talvez e 2=sim
# se a nota da roupa de um dos dois for menor ou igual a 2
# vocês não terão direito à uma mesa (0)
# se as notas são maiores, então caso um dos dois esteja
# bem chique (nota >= 8) então a resposta é sim (2)
# caso contrário a resposta é talvez (1)
# date_fashion(5, 10) -> 2
# date_fashion(5, 2) -> 0
# date_fashion(5, 5) -> 1
def date_fashion(eu, par):
  if eu <= 2 or par <= 2:
    return 0
  if eu >= 8 or par >= 8:
    return 2
  return 1

# H. squirrel_play
```

```python
# A. first_last6
# verifica se 6 é o primeiro ou último elemento da lista nums
# first_last6([1, 2, 6]) -> True
# first_last6([6, 1, 2, 3]) -> True
# first_last6([3, 2, 1]) -> False
def first_last6(nums):
  return nums[0] == 6 or nums[-1] == 6

# B. same_first_last
# retorna True se a lista nums possui pelo menos um elemento
# e o primeiro elemento é igual ao último
# same_first_last([1, 2, 3]) -> False
# same_first_last([1, 2, 3, 1]) -> True
# same_first_last([1, 2, 1]) -> True
def same_first_last(nums):
  return len(nums) > 0 and nums[0] == nums[-1]

# C. common_end
# Dada duas listas a e b verifica se os dois primeiros são
# iguais ou os dois últimos são iguais
# suponha que as listas tenham pelo menos um elemento
# common_end([1, 2, 3], [7, 3]) -> True
# common_end([1, 2, 3], [7, 3, 2]) -> False
# common_end([1, 2, 3], [1, 3]) -> True
def common_end(a, b):
  return a[0] == b[0] or a[-1] == b[-1]

# D. maior_ponta
# Dada uma lista não vazia, cria uma nova lista onde todos
# os elementos são o maior das duas pontas
# obs.: não é o maior de todos, mas entre as duas pontas
# maior_ponta([1, 2, 3]) -> [3, 3, 3]
# maior_ponta([1, 3, 2]) -> [2, 2, 2]
def maior_ponta(nums):
  return [max(nums[0], nums[-1])] * len(nums)

# I. sem_pontas
# seja uma string s de pelo menos dois caracteres
# retorna uma string sem o primeiro e último caracter
# without_end('Hello') -> 'ell'
# without_end('python') -> 'ytho'
# without_end('coding') -> 'odin'
def sem_pontas(s):
  return s[1:-1]

# J. roda2
# rodar uma string s duas posições
# a string possui pelo menos 2 caracteres
# left2('Hello') -> 'lloHe'
# left2('Hi') -> 'Hi'
def roda2(s):
  return s[2:] + s[:2]
```

```python
# E. hello_name
# seja uma string name
# hello_name('Bob') -> 'Hello Bob!'
# hello_name('Alice') -> 'Hello Alice!'
# hello_name('X') -> 'Hello X!'
def hello_name(name):
  return f'Hello {name}!'

# F. make_tags
# make_tags('i', 'Yay'), '<i>Yay</i>'
# make_tags('i', 'Hello'), '<i>Hello</i>'
# make_tags('cite', 'Yay'), '<cite>Yay</cite>'
def make_tags(tag, word):
  return f'<{tag}>{word}</{tag}>'

# G. extra_end
# seja um string s com no mínimo duas letras
# retorna três vezes as duas últimas letras
# extra_end('Hello'), 'lololo'
# extra_end('ab'), 'ababab'
# extra_end('Hi'), 'HiHiHi'
def extra_end(s):
  return 3 * s[-2:]

# H. first_half
# seja uma string s
# retorna a primeira metade da string
# first_half('WooHoo') -> 'Woo'
# first_half('HelloThere') -> 'Hello'
# first_half('abcdef') -> 'abc'
def first_half(s):
  return s[:len(s)//2]
```

```python
# A. multstring
# seja uma string s e um inteiro positivo n
# retorna uma string com n cópias da string original
# multstring('Hi', 2) -> 'HiHi'
def multstring(s, n):
  return n * s

# B. string_splosion
# string_splosion('Code') -> 'CCoCodCode'
# string_splosion('abc') -> 'aababc'
# string_splosion('ab') -> 'aab'
def string_splosion(s):
  return ''.join([s[:k] for k in range(len(s) + 1)])
  resp = ''
  for k in range(len(s)):
    resp += s[:k]
  resp += s
  return resp
  k = 1
  resp = ''
  while k <= len(s):
    resp = resp + s[:k]
    k = k + 1
  return resp

# C. array_count9
# conta quantas vezes aparece o 9 numa lista nums
def array_count9(nums):
  return nums.count(9)

# D. array_front9
# verifica se pelo menos um dos quatro primeiros é nove
# array_front9([1, 2, 9, 3, 4]) -> True
# array_front9([1, 2, 3, 4, 9]) -> False
# array_front9([1, 2, 3, 4, 5]) -> False
def array_front9(nums):
  return 9 in nums[:4]
```

```python
# temos um papagaio que fala alto
# hora é um parâmetro entre 0 e 23
# temos problemas se o papagaio estiver falando
# antes da 7 ou depois das 20
def papagaio(falando, hora):
  return falando and (hora < 7 or hora > 20)

# F. dez
# dados dois inteiros a e b
# retorna True se um dos dois é 10 ou a soma é 10
def dez(a, b):
  return a == 10 or b == 10 or a + b == 10

# G. dista10
# seja um inteiro n
# retorna True se a diferença absoluta entre n e 100 ou n e 200
# for menor ou igual a 10
# dista10(93) -> True
# dista10(90) -> True
# dista10(89) -> False
def dista10(n):
  return abs(n - 100) <= 10 or abs(n - 200) <= 10

# H. apaga
# seja uma string s e um inteiro n
# retorna uma nova string sem a posição n
# apaga('kitten', 1) -> 'ktten'
# apaga('kitten', 4) -> 'kittn'
def apaga(s, n):
  return s[:n] + s[n+1:]

# I. troca
# seja uma string s
# se s tiver tamanho <= 1 retorna ela mesma
# caso contrário troca a primeira e última letra
# troca('code') -> 'eodc'
# troca('a') -> 'a'
# troca('ab') -> 'ba'
def troca(s):
  return s if len(s) <= 1 else s[-1] + s[1:-1] + s[0]
```