Resultado parcial Python IP/Algo1

Parcial - tema1

Nota: 7.6 / 10.0 (APROBADO)

```
puntaje ej1: 2
puntaje ej2: 2
puntaje ej3: 3
puntaje ej4: 0.6
```

Datos del alumno

Nombre: Tomas Mateos

Maquina: 42-03

Enunciado

Introducción a la Programación

Segundo Parcial

- El parcial se aprueba con 6 puntos
- Utilizar [/parcial/solucion_t1.py]este archivo fuente de base para la programación. Ya cuenta con los def y las signaturas correctas.
- Para testear el código pueden usar [/parcial/test-t1.py]este archivo que ya cuenta con todo lo necesario para desarrollar sus propios tests (este archivo no se entrega)
- **Para aprobar el parcial es requisito indispensable que todos los programas pasen los tests del archivo del punto anterior**

---

1) Intercalar [2 puntos]

A la hora de jugar juegos de cartas, como el truco, el tute, o el chinchón, es importante que la distribución de las mismas en el mano sea aleatoria. Para esto, al comienzo de cada mano, antes de repartir las mismas se realizan mezclan sucesivas. Una técnica de mezclado es la denominada

"mezcla americana" que consiste en separar el mazo en (aproximadamente) dos mitades e intercalar las cartas de ambas mitades. Implementar la función intercalar que dadas dos listas *s1* y *s2* con igual cantidad de elementos devuelva una lista con los elementos intercalados. Esto es, las posiciones pares de *res* tendrán los elementos de *s1* y las posiciones impares los elementos de *s2*, respetando el orden original.

problema intercalar (in *s1*: seq⟨Z⟩, in *s2*: seq⟨Z⟩) : seq⟨Z⟩ {
  requiere: {|*s1*| = |*s2*| }
  asegura: {|*res*| = 2 * |*s1*|}
  asegura: {*res*[2*i] = *s1*[i] y *res*[2*i+1] = *s2*[i], para i entre 0 y |*s1*|-1}
}

TIP: realizar la iteración mediante índices y no mediante elementos

Por ejemplo, dadas
s1 = [1, 3, 0, 1]
s2 = [4, 0, 2, 3]
se debería devolver *res* = [1, 4, 3, 0, 0, 2, 1, 3]

---

2) Posición de n-ésima aparición [2 puntos]

Guido y Marcela son dos estudiantes de IP, nervioses con el parcial de Python. Con el objetivo de tener un rato antes del parcial para preguntarse algunas dudas deciden encontrarse en el colectivo y viajar juntes. Para poder coordinar de forma exacta en qué colectivo se tienen que subir, Marcela usa sus habilidades de programación aprendidas en IP para acceder de forma poco legítima a la base de datos de colectivos de todas las empresas. Con esto, arma una lista de todos los colectivos que van a pasar por la parada de Guido alrededor del horario acordado y le indica a Guido que se tiene que subir en el 3er colectivo de la línea 34. Por desgracia, Guido se olvida sus lentes antes de salir y no es capaz de distinguir a qué línea pertenece cada colectivo que llega a la parada. Por lo que solo puede contar cantidad total de colectivos que pasan.

Implementar la función pos_nesima_aparicion que dada una secuencia de enteros *s*, y dos enteros *n* y *elem* devuelve la posición en la cual *elem* aparece por *n*-ésima vez en *s*. En caso de que *elem* aparezca menos de *n* veces en *s*, devolver -1.

problema pos_nesima_aparicion (in *s*: seq⟨Z⟩, in *n*: Z, in *elem*: Z) : Z {
  requiere: {n>0}
  asegura: {Si el elemento aparece menos de *n* veces en la secuencia, *res*= -1 }
  asegura: {Si el elemento aparece al menos *n* veces en la secuencia, *s*[*res*] = *elem* }
  asegura: {Si el elemento aparece al menos *n* veces en la secuencia, *elem* aparece *n-1* veces en *s* entre las posiciones 0 y *res-1* }
}

Por ejemplo, dadas
s = [-1, 1, 1, 5, -7, 1, 3]
n = 2
elem = 1
se debería devolver *res* = 2

3) Matriz espejada [3 puntos]

Implementar la función matriz_espejada que dada una matriz devuelve True si cada una de sus filas es capicúa. Es decir, si cada fila es igual leída de izquierda a derecha o de derecha a izquierda. Definimos a una secuencia de secuencias como matriz si todos los elemento de la primera secuencia tienen la misma longitud.

problema matriz_espejada(in *m*:seq⟨seq⟨Z⟩⟩) : Bool {
  requiere: {todos los elementos de *m* tienen igual longitud (los elementos de *m* son secuencias)}
  asegura: {(*res* = true) <=> todos los elementos de *m* son capicúa}
}

Por ejemplo, dada la matriz
m = [[1,2,2,1],[-5,6,6,-5],[0,1,1,0]]
se debería devolver *res* = true

4) En el hipódromo: posiciones de los caballos [3 puntos]

Además de recitales de artistas de renombre internacional (ej: Bizarrap), en el hipódromo de Palermo se realizan cotidianamente carreras de caballos. Por ejemplo, durante el mes de Octubre se corrieron 10 carreras. En cada una de ellas participaron alrededor de 10 caballos.

Implementar la función cuenta_posiciones_por_caballo que dada la lista de caballos que corrieron las carreras, y el diccionario que tiene los resultados del hipódromo en el formato carreras:posiciones_caballos, donde carreras es un String y posiciones_caballos es una lista de strings con los nombres de los caballos, genere un diccionario de caballos:#posiciones, que para cada caballo devuelva la lista de cuántas veces salió en esa posición.

Tip: para crear una lista con tantos ceros como caballos se puede utilizar la siguiente sintaxis
lista_ceros = [0]*len(caballos)

problema cuenta_posiciones_por_caballo(in *caballos*: seq⟨String⟩, in *carreras*: dict⟨String,seq⟨String⟩⟩: dict⟨String,seq⟨Z⟩⟩ {
  requiere: {*caballos* no tiene repetidos}
  requiere: {Los valores del diccionario *carreras* son permutaciones de la lista *caballos* (es decir, tienen exactamente los mismos elementos que *caballos*, en cualquier orden posible)}
  asegura: {*res* tiene como claves los elementos de *caballos*}
  asegura: {El valor en *res* de un caballo es una lista que indica en la posición *i* cuántas veces salió ese caballo en la *i*-ésima posición.}
}

Por ejemplo, dados
caballos= ["linda", "petisa", "mister", "luck"]
carreras= {"carrera1":["linda", "petisa", "mister", "luck"],
"carrera2":["petisa", "mister", "linda", "luck"]}

se debería devolver *res* = {"petisa": [1,1,0,0],
"mister": [0,1,1,0],
"linda": [1,0,1,0],
"luck": [0,0,0,2]}

Solucion entregada por el alumno

```
# Ejercicio 1
def intercalar(s1: list, s2: list) -> list:
    res:list=[]
    for i in range(len(s1)):
        res = res + [s1[i],s2[i]]
    return res

# la funcion toma el primer elemento de la lista 2 y el primero de la lista 1,
en ese orden y los va sumando a una lista aparte.


# Ejercicio 2
def pos_nesima_aparicion(s: list, n: int, elem: int) -> int:
    cont:int=0
    for i in range(len(s)):
        if elem==s[i]:
            cont+=1
            if n==cont:
                return i
    return -1
# la funcion utiliza un contador para saber cuantas veces aparecio el elemento.
cuando el numero de apariciones en la lista
# es el mismo que el ingresado, devuelve el indice donde aparecio el elemento.
si no aparece o no aparece las veces que se pide
# entonces devuelve -1

# Ejercicio 3
def matriz_espejada(m: list) -> bool:
    for i in range(len(m)):
        if not igualrevez(m[i]):
            return False
    return True
```

```
# la funcion utiliza la funcion auxiliar para ver si alguna lista de la matriz
no es espejada. si alguna no es espejada devuelve False
# sino devuelve True

#funcion auxiliar ej 3
def igualrevez(lista:[int])->bool:
    for i in range(len(lista)):
        j=(len(lista)-i)-1
        if lista[i]!=lista[j]:
            return False
    return True
# checkea si el primer elemento es igual al ultimo, si el segundo igual al
anteultimo y asi.
# si ocurre que dos elementos no son iguales entonces devuelve false.

# Ejercicio 4
def cuenta_posiciones_por_caballo(caballos: list, carreras: dict) -> dict:
    res:dict={}
    for caballo in caballos:
        res.update({caballo:[0]*len(caballos)})
    return res
# no me acuerdo como acceder a la lista dentro del diccionario.
```

Resultado de la compilacion

Ejecucion de los tests

tema1-test-ej1.py.compilacion.out

Puntaje del ej: 2 / 2

```
test_listasDistintas (__main__.Ej1Test) ... ok
test_listasIgualesDeApares (__main__.Ej1Test) ... ok
test_listasIgualesTresElem (__main__.Ej1Test) ... ok
test_listasLargas (__main__.Ej1Test) ... ok
test_listasTresElemIguales (__main__.Ej1Test) ... ok
test_listasUnElemDistintos (__main__.Ej1Test) ... ok
test_listasUnElemIguales (__main__.Ej1Test) ... ok
test_listasVacias (__main__.Ej1Test) ... ok

----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
```

tema1-test-ej2.py.compilacion.out

Puntaje del ej: 2 / 2

```
test_elementoNoEsta (__main__.Ej2Test) ... ok
test_elementoNoEstaNveces (__main__.Ej2Test) ... ok
test_elementoNoEstaNvecesBis (__main__.Ej2Test) ... ok
test_listaVacia (__main__.Ej2Test) ... ok
test_masDeNAparicionesConsecutivas (__main__.Ej2Test) ... ok
test_masDeNAparicionesIntercalado (__main__.Ej2Test) ... ok
test_primeraAparicionAlFinal (__main__.Ej2Test) ... ok
test_primeraAparicionAlPpio (__main__.Ej2Test) ... ok
test_primeraAparicionEnElMedio (__main__.Ej2Test) ... ok
test_segundaAparicionAlFinal (__main__.Ej2Test) ... ok
test_segundaAparicionEnElMedio (__main__.Ej2Test) ... ok
test_todosIgualesApIntermedia (__main__.Ej2Test) ... ok
test_todosIgualesNoHayNAparicion (__main__.Ej2Test) ... ok
test_todosIgualesPrimeraAp (__main__.Ej2Test) ... ok
test_todosIgualesUltimaAparicion (__main__.Ej2Test) ... ok

----------------------------------------------------------------------
Ran 15 tests in 0.001s

OK
```

tema1-test-ej3.py.compilacion.out

Puntaje del ej: 3 / 3

```
test_MedioNoEspejad0 (__main__.Ej3Test) ... ok
test_UltimaColNoEspejada (__main__.Ej3Test) ... ok
test_UltimaFilaNoEspejada (__main__.Ej3Test) ... ok
test_listaVacia (__main__.Ej3Test) ... ok
test_matrizDeCerosFilasImpares (__main__.Ej3Test) ... ok
test_matrizDeCerosFilasPares (__main__.Ej3Test) ... ok
test_matrizEspejada (__main__.Ej3Test) ... ok
test_matrizEspejadaBis (__main__.Ej3Test) ... ok
test_primeraFilaEspejada (__main__.Ej3Test) ... ok
test_primeraFilaNoEspejada (__main__.Ej3Test) ... ok
test_unaColumnaDistinta (__main__.Ej3Test) ... ok
test_unaColumnaTodosIguales (__main__.Ej3Test) ... ok
test_unaFilaUnElem (__main__.Ej3Test) ... ok

----------------------------------------------------------------------
Ran 13 tests in 0.000s

OK
```

tema1-test-ej4.py.compilacion.out

Puntaje del ej: 0.6 / 3

```
test_carrerasSinCaballos (__main__.Ej4Test) ... ok
```

```
test_dosAlaCabezaDosAlFinal (__main__.Ej4Test) ... FAIL
test_enunciado (__main__.Ej4Test) ... FAIL
test_muchosCaballosCarrerasIguales (__main__.Ej4Test) ... FAIL
test_muchosCaballosGananLoMismo (__main__.Ej4Test) ... FAIL
test_muchosCaballosUnaCarrera (__main__.Ej4Test) ... FAIL
test_todoVacio (__main__.Ej4Test) ... ok
test_unCaballoMuchasCarreras (__main__.Ej4Test) ... FAIL
test_unCaballoUnaCarrera (__main__.Ej4Test) ... FAIL
test_variasCarrerasDistintoOrden (__main__.Ej4Test) ... FAIL

======================================================================
FAIL: test_dosAlaCabezaDosAlFinal (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 82, in
test_dosAlaCabezaDosAlFinal
    self.assertDictEqual({"petisa": [2,1,0,0], "mister": [0,0,2,1], "linda":
[1,2,0,0], "luck": [0,0,1,2]}, res)
AssertionError: {'petisa': [2, 1, 0, 0], 'mister': [0, 0, 2[45 chars], 2]} !=
{'linda': [0, 0, 0, 0], 'petisa': [0, 0, 0,[45 chars], 0]}
- {'linda': [1, 2, 0, 0],
+ {'linda': [0, 0, 0, 0],
-  'luck': [0, 0, 1, 2],
?                 ^  ^

+  'luck': [0, 0, 0, 0],
?                 ^  ^

-  'mister': [0, 0, 2, 1],
?                   ^  ^

+  'mister': [0, 0, 0, 0],
?                   ^  ^

-  'petisa': [2, 1, 0, 0]}
?             ------

+  'petisa': [0, 0, 0, 0]}
?               ++++++


======================================================================
FAIL: test_enunciado (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 74, in
test_enunciado
    self.assertDictEqual({"petisa": [1,1,0,0], "mister": [0,1,1,0], "linda":
[1,0,1,0], "luck": [0,0,0,2]}, res)
AssertionError: {'petisa': [1, 1, 0, 0], 'mister': [0, 1, 1[45 chars], 2]} !=
{'linda': [0, 0, 0, 0], 'petisa': [0, 0, 0,[45 chars], 0]}
```

```
- {'linda': [1, 0, 1, 0],
?            ---   ^

+ {'linda': [0, 0, 0, 0],
?               ^ +++

-  'luck': [0, 0, 0, 2],
?                   ^

+  'luck': [0, 0, 0, 0],
?                   ^

-  'mister': [0, 1, 1, 0],
?                ^  ^

+  'mister': [0, 0, 0, 0],
?                ^  ^

-  'petisa': [1, 1, 0, 0]}
?             ------

+  'petisa': [0, 0, 0, 0]}
?               ++++++


======================================================================
FAIL: test_muchosCaballosCarrerasIguales (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 61, in
test_muchosCaballosCarrerasIguales
    self.assertDictEqual({"linda": [3,0,0], "buena": [0,3,0], "pipi": [0,0,3]},
res)
AssertionError: {'linda': [3, 0, 0], 'buena': [0, 3, 0], 'pipi': [0, 0, 3]} !=
{'linda': [0, 0, 0], 'buena': [0, 0, 0], 'pipi': [0, 0, 0]}
- {'buena': [0, 3, 0], 'linda': [3, 0, 0], 'pipi': [0, 0, 3]}
?               ^                ---                       ^

+ {'buena': [0, 0, 0], 'linda': [0, 0, 0], 'pipi': [0, 0, 0]}
?               ^                  +++                     ^


======================================================================
FAIL: test_muchosCaballosGananLoMismo (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 67, in
test_muchosCaballosGananLoMismo
    self.assertDictEqual({"linda": [1,1,1], "buena": [1,1,1], "pipi": [1,1,1]},
res)
AssertionError: {'linda': [1, 1, 1], 'buena': [1, 1, 1], 'pipi': [1, 1, 1]} !=
```

```
{'linda': [0, 0, 0], 'buena': [0, 0, 0], 'pipi': [0, 0, 0]}
- {'buena': [1, 1, 1], 'linda': [1, 1, 1], 'pipi': [1, 1, 1]}
?            ^  ^  ^             ^  ^  ^            ^  ^  ^

+ {'buena': [0, 0, 0], 'linda': [0, 0, 0], 'pipi': [0, 0, 0]}
?            ^  ^  ^             ^  ^  ^            ^  ^  ^


======================================================================
FAIL: test_muchosCaballosUnaCarrera (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 55, in
test_muchosCaballosUnaCarrera
    self.assertDictEqual({"linda": [1,0,0], "buena": [0,1,0], "pipi": [0,0,1]},
res)
AssertionError: {'linda': [1, 0, 0], 'buena': [0, 1, 0], 'pipi': [0, 0, 1]} !=
{'linda': [0, 0, 0], 'buena': [0, 0, 0], 'pipi': [0, 0, 0]}
- {'buena': [0, 1, 0], 'linda': [1, 0, 0], 'pipi': [0, 0, 1]}
?               ^                ---                        ^

+ {'buena': [0, 0, 0], 'linda': [0, 0, 0], 'pipi': [0, 0, 0]}
?               ^                  +++                      ^


======================================================================
FAIL: test_unCaballoMuchasCarreras (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 49, in
test_unCaballoMuchasCarreras
    self.assertDictEqual({"linda": [3]}, res)
AssertionError: {'linda': [3]} != {'linda': [0]}
- {'linda': [3]}
?            ^

+ {'linda': [0]}
?            ^


======================================================================
FAIL: test_unCaballoUnaCarrera (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 43, in
test_unCaballoUnaCarrera
    self.assertDictEqual({"linda": [1]}, res)
AssertionError: {'linda': [1]} != {'linda': [0]}
- {'linda': [1]}
?            ^
```

```
+ {'linda': [0]}
?            ^


======================================================================
FAIL: test_variasCarrerasDistintoOrden (__main__.Ej4Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/pablo/git/intro-programacion/2c2023/examen_online/correcciones
/2parcial-tm/mateos_tomas_44115190/tema1-test-ej4.py", line 89, in
test_variasCarrerasDistintoOrden
    self.assertEqual(resultado, {"trueno":[1,2,0], "remolino":[1,1,1], "crack":
[1,0,2]})
AssertionError: {'trueno': [0, 0, 0], 'remolino': [0, 0, 0], 'crack': [0, 0,
0]} != {'trueno': [1, 2, 0], 'remolino': [1, 1, 1], 'crack': [1, 0, 2]}
- {'crack': [0, 0, 0], 'remolino': [0, 0, 0], 'trueno': [0, 0, 0]}
?               ^^^^                ^  ^  ^              ^  ^

+ {'crack': [1, 0, 2], 'remolino': [1, 1, 1], 'trueno': [1, 2, 0]}
?            +++   ^                ^  ^  ^              ^  ^


----------------------------------------------------------------------
Ran 10 tests in 0.004s

FAILED (failures=8)


FIN
```