Programación—Certamen 2 (CC) - Martes 9 de Junio de 2015

Nombre: Rol: –

1. [25 %] Realice el ruteo de los siguientes programas e indique qué es lo que imprimen. Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla. Recuerde que si una variable es de tipo string, debe colocar su valor entre comillas simples ' '. Si una variable almacena una función, coloque el nombre de ésta como valor (sin comillas).

***Importante**: La tabla tiene suficientes filas.*

```
def f2(L,i):
  L[i+1] = L[i]
  i = i-1
  return L,i

def f1(t):
  for i in range(1,len(t)):
    val = t[i]
    j = i - 1
    while j>=0 and t[j]>val:
      t,j = f2(t,j)
    t[j+1] = val
  return t

L = [6,3,4]
L = f1(L)
print L
```

global	f1				f2	
L	t	i	val	j	L	i

Análisis de algoritmo

Analice el siguiente algoritmo y determine, en pocas palabras, lo que realiza. No debe usar más del espacio indicado.

```
# n numero entero
def f(n):
    l = []
    for i in range(len(str(n))-1,-1,-1):
        x = n/(10**i)
        l.append(x)
        n=n%(10**i)
    return l
```

Programación—Certamen 2 (CC) - Martes 9 de Junio de 2015

Nombre: Rol: –

2. **[35 %]** El prestamista Vito Corleone, aburrido de perseguir a sus clientes, ha solicitado la ayuda de algún astro de la programación para resolver sus problemas logísticos.

Vito cuenta con la lista `clientes`, la que tiene tuplas con el nombre de cada cliente, el monto adeudado y la fecha del último pago (también dentro de una tupla).

```
clientes = [('Don Ramon', 3500, (9, 4, 2014)), ('Miguel', 2785, (30,
   10, 2014)), ('Cesar', 100, (28, 5, 2015)), # ...
]
```

Nota: Esto es sólo un ejemplo, considere que la lista puede tener muchos clientes. Sin embargo, asuma que cada nombre de cliente aparece sólo una vez.

a) Implemente la función `deuda_total(clientes)`, que reciba como entrada la lista de tuplas `clientes`, y retorne la suma de las deudas de todos los deudores.

```
>>> deuda_total(clientes)
6385
```

b) Implemente la función `mayor_deudor(clientes, ultimo)`, que retorne el nombre del cliente que tiene la mayor deuda, y que además su último pago haya sido realizado en el año `ultimo`. Si no hay clientes con pagos en el año `ultimo`, retorne un string vacío.

```
>>> mayor_deudor(clientes, 2014)
'Don Ramon'
```

c) Implemente la función `pagar(clientes, pago)` que recibe la lista `clientes` y una tupla `pago` compuesta por el nombre de quien paga, el monto que cancela y la fecha en que lo hace. La función **modifica y retorna** la lista `clientes`, descontando la deuda y actualizando la fecha del último pago, según se indique en la tupla `pago`. Si la deuda llega a 0, debe borrar al cliente de esta lista. No es necesario considerar el caso en que se paga más de lo que se debe.

```
>>> pagar(clientes, ('Miguel', 85, (3, 6, 2015)))
[('Don Ramon', 3500, (9, 4, 2014)), ('Miguel', 2700, (3, 6, 2015)), ('
   Cesar', 100, (28, 5, 2015))]
>>> pagar(clientes, ('Cesar', 100, (4, 6, 2015)))
[('Don Ramon', 3500, (9, 4, 2014)), ('Miguel', 2700, (3, 6, 2015))]
```

Programación—Certamen 2 (CC) - Martes 9 de Junio de 2015

Nombre: _____ Rol: _____ – _

3. **[40 %]** El villano más malvado sobre la faz de la tierra, Ultron, tiene catalogado a todos los superhéroes de la tierra de la siguiente forma:

```
# nombre: [detalle, edad, habilidad, (min-poder, max-poder)]
superheroes = {
'Iron man': ['mk42', 50 , 'uni-rayo',  (45, 95)],
'Thor': ['hijo de odin', 10000 , 'mjolnir',  (50, 100)],
'Condorito':['de pelotillehue', 40, 'washington', (1, 10)],
'Chapulin Colorado': ['no contaban con mi astucia', 40, 'chipote
   chillon', (40, 90)],
# ...
}
# nombre : nombre_asociados
asociados = {
'Iron man': set(['Thor', 'Black Widow', 'Hawkeye', 'Hulk']),
'Thor':  set(['Iron man', 'Hulk', 'Chapulin Colorado']),
'Condorito' : set(['Don Chuma', 'Hulk']),
'Chapulin Colorado' : set(['Condorito', 'Thor']),
# ...
}
```

Pese a ser la inteligencia artificial más avanzada jamás creada, se le complica un poco cuando llega el momento de programar, por ello pide ayuda a los alumnos de IWI-131 en lo siguiente:

a) Desarrollar la función `diferencias_poder(superheroes, diferenciapoder, umbral)` que reciba el diccionario `superheroes`, el valor `diferenciapoder` y un valor de `umbral`. La función debe retornar una lista de tuplas de todos los superhéroes que tengan una diferencia de poder (diferencia min-poder y max-poder) mayor o igual a `diferenciapoder` y un min-poder superior al `umbral`. Cada tupla debe contener nombre, detalle, habilidad y max-poder.

```
>>> diferencias_poder(superheroes, 30, 39)
[('Iron man', 'mk42', 'uni-rayo', 95), ('Thor', 'hijo de odin', '
   mjolnir', 100), ('Chapulin Colorado', 'no contaban con mi astucia',
   'chipote chillon', 90)]
```

b) Ultron necesita saber quién es amigo de quien. Dos asociados se consideran amigos si cada uno tiene al otro en el diccionario asociados. Genere una función `amigos(asociados)` que reciba el diccionario `asociados` y retorne un diccionario con los amigos, donde la llave es el superhéroe y como valor un conjunto con los amigos de éste. Si un superhéroe no tiene amigos, no se agrega.

```
>>> amigos(asociados)
{'Iron man': set(['Thor']), 'Chapulin Colorado': set(['Thor']), 'Thor':
    set(['Iron man', 'Chapulin Colorado'])}
```

c) Gracias a la manipulación mental de Scarlet Witch, Ultron logrará que los superhéroes amigos que posean una diferencia de poder superior a 40 y además que superen un umbral de min-poder de 30 luchen entre ellos. Se pide generar una función `versus(superheroes, asociados)` que reciba el diccionario `superheroes`, el diccionario `asociados` y retorne una lista de tuplas con los enfrentamientos de los amigos. NOTA: no se deben repetir las parejas.

```
>>> versus(supeheroes, asociados)
[('Iron man', 'Thor'), ('Chapulin Colorado', 'Thor')]
```