Programación—Certamen 2 (CC) - Lunes 21 de Diciembre de 2015

Nombre: ______________________ Rol: __________ – __

1. **[20 %]** 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 f1(datos):
    for i in range(1, len(datos)):
        j = i
        while j > 0 and datos[j] < datos[j-1]:
            temp = datos[j]
            datos[j] = datos[j-1]
            datos[j-1] = temp
            j -= 1
    return datos

lista = [4, 3, 6, 1]
lista = f1(lista)
print lista[::-1]
```

global	f1			
lista	datos	i	j	temp

Programación—Certamen 2 (CC) - Lunes 21 de Diciembre de 2015

Nombre: _____ Rol: _____ – _____

2. **[40 %]** Santa Claus ha comprado un sistema hecho en Python para administrar la Navidad 2015. Básicamente, el sistema maneja una lista de tuplas, donde cada tupla corresponde a los registros de una persona con el formato `(nombre, (x, y), tipo)`, donde `(x, y)` son las coordenadas en el mapa de la ubicación de la persona y `tipo` indicaba como se ha portado la persona (`'B'`: Bueno; `'M'`: Malo). Ver **ejemplo**:

```
personas = [('Erick Lopez', (12.9887, 1.5567), 'M'), ('Cesar Moltedo',
   (12.1986, 2.5321), 'M'), ('Andrew Ng', (-2.9001, 7.6453), 'M'), ('
   Zafradaa', (12.2316, 5.0089), 'B'), # ... ]
```

Nota: Asuma que cada nombre aparece sólo una vez.

Ahora usted debe:

a) Desarrolle la función `persona_mas_cerca(personas, posicion)` que reciba la lista `personas` y la posición actual (`posicion`) de Santa Claus. La función debe retornar el nombre de la persona más cercana a la posición de Santa Claus. Para calcular esta distancia recuerde que, en $\mathbb{R}^2$, si se tiene un punto (x_1, y_1) y un punto (x_2, y_2), entonces la distancia entre los dos puntos viene dada por $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

```
>>> persona_mas_cerca(personas, (12.9676, 1.4991))
'Erick Lopez'
```

b) Santa Claus ha determinado que la mejor ruta para repartir todos los regalos es ir siempre a la persona más cercana (con cuidado de no repetirlas) desde la posición en la que se encuentra. Usted debe crear la función `mejor_ruta(personas, posicion)` que reciba la lista `personas` y la posición inicial de Santa Claus, y retorne una lista con la ruta a seguir identificada por el nombre de las personas.

Nota: No altere la lista original de `personas`.

```
>>> mejor_ruta(personas, (0.0, 0.0))
['Andrew Ng', 'Zafradaa', 'Cesar Moltedo', 'Erick Lopez']
```

c) A pocos días de la Navidad, Santa Claus se ha dado cuenta que la lista con `personas` posee errores ya que el coordinador de programación no puede ser una persona mala. Gracias a esta observación, Santa concluyó que el Grinch había hackeado el sistema alterando el campo `tipo` de algunas personas, es decir, una persona buena fue cambiada a mala o vise-versa. Por suerte Santa tiene un respaldo, una lista con el nombre y el tipo de cada persona. Ver **ejemplo**:

```
respaldo = [('Cesar Moltedo', 'B'), ('Zafradaa', 'B'), ('Andrew Ng', 'B
   '), ('Erick Lopez', 'M')]
```

Ahora usted debe implementar una función `restaurar(personas, respaldo)` que reciba como parámetro las listas `personas` y `respaldo`. La función debe restaurar la lista `personas`, es decir, volver el `tipo` de cada persona (según `respaldo`). La función retorna **`None`**.

```
>>> restaurar(personas, respaldo)
>>> print personas
[('Erick Lopez', (12.9887, 1.5567), 'M'), ('Cesar Moltedo', (12.1986,
   2.5321), 'B'), ('Andrew Ng', (-2.9001, 7.6453), 'B'),  ('Zafradaa',
   (12.2316, 5.0089), 'B')]
```

Programación—Certamen 2 (CC) - Lunes 21 de Diciembre de 2015

Nombre: _____ Rol: _____ – _____

3. **[40 %]** La plataforma de citas en línea MatchMaker ha decidido contratarlo para ayudarles en el análisis de su base de datos. Por suerte, la base de datos corresponde a una lista de diccionarios de python como el siguiente:

```
MatchMaker = [
{'nombre': 'Sheldon Cooper', 'sexo': 'M', 'pasatiempos': {'Sheldon', '
   ciencia', 'fisica', 'trenes'}},
{'nombre': 'Leonard Hofsadter', 'sexo': 'M', 'pasatiempos': {'fisica',
   'comics'}},
{'nombre': 'Howard Wolowitz', 'sexo': 'M', 'pasatiempos': {'ingenieria'
    , 'comics'}},
{'nombre': 'Raj Koothrappali', 'sexo' :'M', 'pasatiempos': {'astronomia
   ', 'comics'}},
{'nombre': 'Penny', 'sexo':'F', 'pasatiempos': {'ciencia', 'peliculas',
   'compras'}},
{'nombre': 'Bernadette Rostenkowski', 'sexo': 'F', 'pasatiempos': {'
   biologia', 'compras'}},
{'nombre': 'Amy Farrah Fowler', 'sexo': 'F', 'pasatiempos': {'ciencia',
    'fisica', 'biologia', 'Sheldon'}},
# ...
]
```

Obviamente, la base de datos real tiene muchos más datos. Sin embargo, puede asumir que cada nombre es único en la base de datos.

Se le pide crear funciones para cumplir con los siguientes requerimientos:

a) Escriba la función `posicion(MatchMaker, nombre)` que indique la posición (índice) en la base de datos de la persona `nombre`. Si el nombre no existe, retornar **`None`**.

```
>>> posicion(MatchMaker, 'Leonard Hofsadter')
1
>>> posicion(MatchMaker, 'Penny')
4
>>> print posicion(MatchMaker, 'Barney Stinson')
None
```

b) Escribe la función `cantidad_pasatiempos(MatchMaker)` que regrese un diccionario con el número de personas por cada pasatiempo que existen en la base de datos.

```
>>> cantidad_pasatiempos(MatchMaker)
{'Sheldon': 2, 'peliculas': 1, 'comics': 3, 'fisica': 3, 'compras': 2,
   'astronomia': 1, 'trenes': 1, 'ingenieria': 1, 'biologia': 2, '
   ciencia': 3}
```

c) Escriba la función `pareja_mas_afin(MatchMaker)` que regrese el par de personas, de sexo opuesto, que más pasatiempos en común tengan. En caso de empate, regrese cualquiera de las parejas empatadas. Si no hay parejas afines, la función retorna **`None`**.

```
>>> pareja_mas_afin(MatchMaker)
('Sheldon Cooper', 'Amy Farrah Fowler')
```