Programación – Certamen 2 - Martes 23 de Julio de 2019

Nombre

Rol — Paralelo Hopcroft

1. [20 %] Realice el ruteo del siguiente programa e indique en el recuadro de la derecha lo que imprime como salida. Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla. Si una variable es de tipo string, su valor debe ir entre comillas simples ' ' o dobles " ". La tabla tiene suficientes filas.

```python
def f(p):
   if '-' in p:
      x = p[1]
   else:
      x = p[3]
   return x

s = ''
for q in ['HBRU 93', 'XS-7122', 'CM-4513']:
   c = f(q)
   s = s + c
print(s)
```

Salida:

programa			f	
s	q	c	p	x

Programación – Certamen 2 - Martes 23 de Julio de 2019

Nombre

Rol – Paralelo Hopcroft

2.

Programación – Certamen 2 - Martes 23 de Julio de 2019

Nombre

Rol – Paralelo Hopcroft

3.

Programación – Certamen 2 - Martes 23 de Julio de 2019

2. **[40 %]** La policía de Pythonia le ha encargado construir un programa para cursar multas a quienes no respeten la restricción vehicular. Para ello se cuenta con la lista `registro`, que almacena los datos de cada vehículo en una tupla que contiene la patente, la ciudad, el dueño, y un booleano que indica si el vehículo es catalítico (`True`) o no lo es (`False`). Por ejemplo:

```
registro = [
   ('CRTJ 32', 'Valpyraiso', 'Juan Perez', True),
   ('RX-2134', 'Sanpyago', 'Ana Martinez', False),
   ('ADNT 28', 'Pyna del Mar', 'Fernanda Jara', False),
   ('ZZ-9999', 'Pyca', 'Pedro Allende', True),
   ('AK-2130', 'Sanpyago', 'Paloma Blanco', True),
   ('ABCD 19', 'Copymbo', 'Ana Roa', False)]
```

Las patentes tienen dos formatos distintos:

- 4 letras, un espacio y 2 dígitos (Ejemplo: `'CRTJ 32'`).
- 2 letras, un guión y 4 dígitos (Ejemplo: `'RX-2134'`).

Para definir la restricción de los **catalíticos** se analiza la última letra de la patente, es decir, la letra que está más a la derecha:

- Lunes: patentes cuya última letra sea menor o igual a `'G'`.
- Miércoles: patentes cuya última letra sea mayor que `'G'` pero menor o igual a `'N'`.
- Viernes: patentes cuya última letra sea mayor que `'N'`.

Para el caso de los **no catalíticos** la restricción se decide en base al último dígito:

- Lunes: Patentes terminadas en `0`, `1`, `2` o `3`.
- Miércoles: Patentes terminadas en `4`, `5` o `6`.
- Viernes: Patentes terminadas en `7`, `8` o `9`.

El resto de los días no hay vehículos con restricción, de ningún tipo.

(a) Escriba la función `tiene_restriccion(registro, patente, dia)` que reciba el registro de vehículos, la patente de un vehículo particular, y el día en que el vehículo transitó (todo en mayúscula), y **retorne** `True` si se le debe cursar un parte o `False` en caso contrario. Puede suponer que `patente` existirá siempre en `registro` y que todos los datos son correctos.
Ejemplo:

```
>>> tiene_restriccion(registro, 'CRTJ 32', 'LUNES')
False
>>> tiene_restriccion(registro, 'ZZ-9999', 'VIERNES')
True
```

(b) Escriba la función `restringidos(registro, dia)` que retorne una lista de tuplas, ordenada por ciudad, con los vehículos que no deben circular el día indicado. Cada tupla debe contener la ciudad y la patente, únicamente. Puede utilizar la función anteriormente escrita.
Ejemplo:

```
>>> listar_restringidos(registro, 'MIERCOLES')
[('Sanpyago', 'AK-2130'), ('Sanpyago', 'RX-2134'), ('Valpyraiso', 'CRTJ 32')]
>>> listar_restringidos(registro, 'LUNES')
[]
```

Programación – Certamen 2 - Martes 23 de Julio de 2019

3. **[40 %]** Para decidir el orden de las presentaciones en una clase, la profesora divide a sus estudiantes en dos grupos, de acuerdo a la prioridad académica. En el primer grupo estarán los estudiantes cuya prioridad esté por debajo del promedio de la prioridad académica de los estudiantes de todo el curso. En el segundo grupo estarán los que tienen prioridad mayor o igual al promedio. Dentro de cada grupo, los estudiantes se ordenarán por edad, de menor a mayor.

La lista `info_personal` contiene los datos de **todos los estudiantes del establecimiento**. Cada tupla en la lista contiene el nombre del estudiante, su RUT y su fecha de nacimiento, en formato `(aaaa,mm,dd)`. Por simplicidad vamos a suponer que los nombres no se repiten. Por ejemplo:

```
info_personal = [ ('Andrea Campos', '17.345.908-7', (1990,4,12)),
                  ('Raquel Salinas', '16.231.998-k', (1987,1,12)),
                  ('Pedro Meza', '12.677.800-3', (1975,10,8)),
                  ('Daniel Varas', '15.123.567-9', (1991,3,30)),
                  ('Sergio Pezoa', '9.990.234-1', (1982,12,24)),
                  ('Alvaro Vasquez', '18.836.902-k',(1988,8,8)),
                ...]
```

La prioridad académica de **todos los estudiantes del establecimiento** se encuentra en la lista `info_academica`, donde cada elemento es una lista con el RUT del estudiante y su prioridad académica. Por ejemplo:

```
info_academica = [ ['17.345.908-7', 6743], ['12.677.800-3', 7990],
                   ['9.990.234-1', 2455],  ['16.231.998-k', 4500],
                   ['15.123.567-9', 5431], ['18.836.902-k', 4998]
                 ...]
```

Para saber cuáles son los estudiantes de la profesora, se cuenta con la lista `curso`, que contiene los números de RUT de los alumnos en el curso. Por ejemplo:

```
curso = [ '17.345.908-7', '9.990.234-1', '16.231.998-k',
          '15.123.567-9','18.836.902-k',
        ...]
```

No existe un orden predefinido en las listas, y no necesariamente habrá correspondencia de índices entre los datos de un mismo estudiante en distintas listas.

Desarrolle un programa que implemente el nuevo procedimiento de división y que muestre los nombres de los estudiantes en el formato que se ilustra en el siguiente ejemplo:

```
Primer Grupo: ['Raquel Salinas', 'Sergio Pezoa']
Segundo Grupo: ['Daniel Varas', 'Andrea Campos', 'Alvaro Vasquez']
```