# Programación—Certamen 3 - 14 de Agosto de 2018

Nombre

Rol — Paralelo

**1.** [**20 %**] César Pyllan, el encantador de perros, mantiene el registro de las mascotas que ha atendido en un archivo cuya estructura es *fecha;nombre;raza;problema;solucion*. Un ejemplo de este archivo sería:

```
01-01-2015;boby;salchicha;muerde;collar de ahorque
02-03-2016;billy;pastor aleman;miedo;toque al costado
10-12-2016;boby;san bernardo;muerde;collar de ahorque
24-01-2017;willson;salchicha;miedo;toque al costado
04-02-2017;willson;san bernardo;miedo;toque al costado
03-02-2018;firulais;labrador;estres;SHHH
03-03-2018;firulais;pequines;estres;SHHH
```

La función *leer_perros(na)* se comporta de la siguiente manera:

```
>>> leer_perros("perros.txt")
{'billy': ['toque al costado'], 'firulais': ['SHHH', 'SHHH'],
'boby': ['collar de ahorque', 'collar de ahorque'],
'willson': ['toque al costado', 'toque al costado']}
```

**Ordenamiento**

Se tiene la función *leer_perros(na)* cuyo código se encuentra desordenado. Debe ordenar las líneas de código e identarlas correctamente.

```
 _,n,r,p,s = l.strip().split(';')
pe[n].append(s)
return pe
a = open(na)
a.close()
def leer_perros(na):
pe[n] = []
for l in a:
if n not in pe:
pe = {}
```

**Análisis de Algoritmo**

Explique en menos de 30 palabras lo que realiza la función anterior:

## Programación—Certamen 3 - 14 de Agosto de 2018

Nombre

Rol — Paralelo

**2.** [40 %] **NOTA:** Puede hacer uso de las funciones definidas en la pregunta 1.

Considerando el contexto de la pregunta anterior (el archivo y la función), César Pyllan le solicita ayuda para implementar las siguientes funciones:

a.- Con el fin de poder clasificar los animales por raza, se le solicita desarrollar la función *leer_razas*(*nombre_archivo*) que recibe un string con el nombre del archivo. Esta función retorna un **diccionario** cuya clave es el nombre de la raza y valor un conjunto con los nombres de los perros que pertenecen a ella. Guíese por el ejemplo.

```
>>> leer_razas("perros.txt")
{'salchicha': set(['boby', 'willson']), 'pequines': set(['firulais']),
'pastor aleman': set(['billy']), 'labrador': set(['firulais']),
'san bernardo': set(['boby', 'willson'])}
```

b.- Nuestro amigo César ha notado que existen perros con el mismo nombre en distintas razas. Sabiendo que los nombres de los perros son únicos, esto solo puede significar que existen mestizos. Desarrolle una función llamada *mestizos*(*raza*1, *raza*2, *nombre_archivo*) que recibe dos razas de perro y el nombre del archivo. Esta función debe retornar un **conjunto** con los perros mestizos de dichas razas. Si no existen perros mestizos para las razas consultadas, la función debe retornar un **conjunto vacio**. Guíese por el ejemplo.

```
>>> mestizos("salchicha","san bernardo","perros.txt")
set(['boby', 'willson'])
>>> mestizos("pequines","labrador","perros.txt")
set(['firulais'])
>>> mestizos("pequines","pastor aleman","perros.txt")
set([])
```

c.- Dado que los casos de perros mestizos aumentan, César debe tener soluciones rápidas. Desarrolle la función *solucion_mestizos*(*raza*1, *raza*2, *nombre_archivo*) que, dadas dos razas y el nombre del archivo de los perros, retorna un diccionario cuya llave es la solución y valor la cantidad de veces recomendada dicha solución, considerando solo a los perros mestizos de las razas consultadas.

```
>>> solucion_mestizos("salchicha","san bernardo","perros.txt")
{'toque al costado': 1, 'collar de ahorque': 1}
>>> solucion_mestizos("pequines","labrador","perros.txt")
{'SHHH': 1}
```

## Programación—Certamen 3 - 14 de Agosto de 2018

Nombre

Rol — Paralelo

**3.** [**40 %**] Considere el cine "El Ciclo", donde el precio de la entrada para ver **cualquier película por primera vez** es de $2000, luego el precio va reduciéndose a la mitad, cada vez que una misma persona ve la misma película, teniendo en cuenta que el precio más bajo a pagar por una entrada es $125.

Para esto, el cine cuenta con un archivo llamado `clientes.txt` como se muestra en el ejemplo, en el que registra a todos sus clientes solo con un nombre y un apellido (separado por espacio).

`clientes.txt`

```
John Programation
David Python
Lucas Caminacielos
...
```

Por cada cliente se tiene un archivo compuesto por el nombre del cliente (en el formato nombre_apellido) y la extensión `.txt`, y cuyo contenido corresponde a cada película que dicha persona ha visto en este cine, donde la primera línea corresponde a la primera película que este cliente vio en este cine, luego la segunda línea corresponde a la segunda película vista y así sucesivamente. A continuación se muestra a modo de ejemplo las películas que ha visto uno de los clientes del ejemplo anterior.

`Lucas_Caminacielos.txt`

```
El origen
Los increibles 2
Dracula
El origen
...
```

Se pide que desarrolle un programa en el cual se solicite como entrada el nombre de cierta película, y muestre por pantalla el total de dinero recaudado por esa película en este cine. El programa debe repetir esta tarea hasta que el nombre ingresado sea "0" (string 0). Además, por cada una de estas películas ingresadas se debe generar un archivo de texto cuyo nombre coincida con el nombre COMPLETO de la película, separado por espacios en blanco (por ejemplo: `El origen.txt`), y como contenido tenga el *nombre completo* de todos los clientes que la han visto al menos una vez (el nombre no debe estar repetido y no importa el orden en que se escriban). **Es importante considerar** que si ningún cliente ha visto la película indicada, el archivo no debe crearse.

Nota: Los puntos suspensivos en los archivos reflejan que pueden existir muchos más datos.