## Programación—Certamen 3 - 16 de Junio de 2016

Nombre

Rol — Paralelo

**1.** **[20 %]** Una consulta médica tiene un archivo `pacientes.txt` con los datos personales de sus pacientes. Cada línea del archivo tiene el rut, el nombre y la edad de un paciente, separados por un `':'`. Una línea del archivo se ve así: `12067539-7:Anastasia Lopez:32`.

Además, cada vez que alguien se atiende en la consulta, la visita es registrada en el archivo `atenciones.txt`, agregando una línea que tiene el rut del paciente, la fecha de la visita (en formato dia-mes-año) y el precio de la atención, también separados por `':'`. Una línea del archivo se ve así: `8015253-1:4-5-2010:69580`.

El médico de la consulta necesita saber los nombres de los pacientes que han sido atendidos en un día específico. A continuación se presentan las líneas de código que resuelven este problema, pero están desordenadas. Usted debe ordenarlas e indentarlas (dejar los espacios correspondientes de python) para que ambas funciones estén correctas.

La función `buscar_nombre(rut)` recibe un `rut` y entrega el nombre de la persona con ese rut (asuma que el rut buscado existe en el archivo correspondiente). La función `pacientes_dia(dia, mes, anio)` recibe la fecha y entrega una lista de los nombres de los pacientes que se atendieron ese día.

```
for line in arch:
arch.close()
def buscar_nombre(rut):
arch = open('pacientes.txt')
return n
if rut == r:
r, n, e = line.strip().split(':')
```

```
fecha = map(str,(dia, mes, anio))
arch = open('atenciones.txt')
if fecha == f:
def pacientes_dia(dia, mes, anio):
lista_n = list()
fecha = '-'.join(fecha)
arch.close()
return lista_n
for line in arch:
r, f, c = line.strip().split(':')
lista_n.append(nombre)
nombre  = buscar_nombre(r)
lista_n = list(set(lista_n))
```

## Programación—Certamen 3 - 16 de Junio de 2016

Nombre

Rol — Paralelo

**2.** **[40 %]** La aplicación PYelper le permite conocer los mejores restaurantes en función de lo que otros usuarios han opinado. La información está almacenada en 2 archivos. En el archivo `estrellas.txt` se indica cuantas estrellas (de 0 a 5) le ha dado un usuario a cierto restaurante siguiendo la estructura: `usuario:restaurante:estrellas`. En el archivo `restaurantes.txt` están listados todos los restaurantes que han sido calificados y a qué comuna pertenencen en el siguiente formato: `restaurante:comuna`.

Los siguientes archivos son un **ejemplo** de lo anterior:

`estrellas.txt`

```
comensal2016:py_resto_bar:4
comensal2016:restoPy:3
gloton23:py_resto_bar:3
comensal2016:los_pysales:5
```

`restaurantes.txt`

```
py_resto_bar:comunaA
restoPy:comunaB
los_pysales:comunaA
```

Al respecto se le solicita:

*a*) Escribir la función `enComuna(archivo, comuna)` que retorna una lista con todos los restaurantes que están en la `comuna` descritos en el `archivo`.

```
>>> enComuna('restaurantes.txt', 'comunaA')
['py_resto_bar', 'los_pysales']
```

*b*) Escribir la función `promedioRestaurante(archivo, restaurante)` que retorna el promedio de estrellas que tiene el `restaurante` descrito en el `archivo`. Asuma que cada `restaurante` ha sido calificado al menos una vez.

```
>>> promedioRestaurante('estrellas.txt', 'py_resto_bar')
3.5
```

*c*) Escribir la función `elMejorDeLaComuna(archivoA, archivoB, comuna)` que retorna el nombre del restaurante con mejor promedio de la `comuna`, considerando que en el `archivoA` están las evaluaciones de los restaurantes y en el `archivoB` las comunas en las que están los restaurantes. Si hay más de un restaurante con la mejor evaluación retornar cualquiera, y si no hay restaurantes en la `comuna` retornar un string vacío (`''`).

```
>>> elMejorDeLaComuna('estrellas.txt', 'restaurantes.txt', 'comunaA')
'los_pysales'
```

## Programación—Certamen 3 - 16 de Junio de 2016

Nombre

Rol — Paralelo

**3.** **[40 %]** Twitter ha demostrado ser un buen termómetro del sentir ciudadano. En algunos casos, puede incluso llegar a predecir el resultado de una elección política. Considere que se tiene un archivo con la información de los candidatos a presidente de Pythonia en el formato `apellido;nombre;coalicion`. Una línea del archivo se ve así: `Netero;Isaac;Nueva asociacion`.

Además se tiene un archivo con todas las publicaciones (*tweets*) en Twitter en el formato `usuario;tweet;fecha`, donde `usuario` es quiene posteó el *tweet*, `tweet` el texto del *tweet* y `fecha` es cuando se publicó el *tweet* en formato `AAAA-MM-DD`. Asuma que no hay `';'` dentro del texto o en el nombre de usuario. Una línea del archivo se ve así: `@killua;que voto @presiNetero o freecss;2015-08-17`.

Escriba una función `separa_tweets(arch_tweets, arch_candi)` donde `arch_tweets` corresponde al archivo de tweets y `arch_candi` al archivo de candidatos. La función debe generar un archivo por cada candidado. El nombre del archivo debe ser `apellido.txt`, donde `apellido` es el apellido del candidato en minúscula. El archivo, para cada candidato, debe incluir aquellos *tweets* (con sus tres campos), en los que el texto del *tweet* **incluya** al apellido del candidato, **sin importar si está en mayúsculas o minúsculas**. Un *tweet* puede hacer referencia a más de un candidato, en cuyo caso dicho *tweet* debe ser escrito en cada uno de los archivos de los candidatos mencionados. El archivo debe tener los *tweets* ordenados por fecha de emisión, desde el más reciente al más antiguo en el formato `usuario;tweet;fecha`.