Programación—Certamen 3 - 5 de Julio de 2017

Nombre

Rol — Paralelo

1. [20 %] En la empresa PyPC se tienen los datos de las ventas separados en dos archivos diferentes. El primero `ventas_cliente.txt` contiene el RUT del cliente y una lista de identificadores de ventas en el formato "`RUT/id_venta1,id_venta2,...`". Y el segundo archivo de texto `ventas.txt` contiene una descripción de cada venta con el formato "`id_venta/valor`".
Se muestra ejemplos de ambos archivos:

`ventas_clientes.txt`

```
7789032-1/1,2,4
15231890-4/3
```

`ventas.txt`

```
1/500
2/200
4/800
3/100
```

La tarea principal la realiza la función `crear_arc(a_cli, a_vent, rut)` que recibe el nombre del archivo de clientes, de ventas y un rut, y crea un archivo con nombre `rut.txt` que contiene la siguiente información en cada línea: "`id_venta/valor`" y al final del archivo esté la suma de todas las ventas de dicho cliente..
Como ayuda para la tarea principal se tiene la función `buscar_monto(venta, a_vent)` que al darle un id de venta , y el nombre del archivo de ventas retorna el string "`monto`", que representa el monto de esa venta.

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.
NOTA: los puntos suspensivos denotan la existan de un número variable de datos.

```
return datos[1]
arch = open(arch_vent)
datos = linea.strip().split('/')
def buscar_monto(venta, a_vent):
arch.close()
for linea in arch:
if venta == datos[0]:
```

```
arch.close()
arc2.write(s.format(venta, monto))
r, ventas = linea.strip().split('/')
total = 0
def crear_arc(a_cli, a_vent, rut):
arc = open(a_cli)
monto = buscar_monto(venta, a_vent)
total += int(monto)
arc2 = open(rut + '.txt', 'w')
for venta in l_ventas:
for linea in arc:
s = '{}-{}\n'
if r == rut:
arc2.close()
l_ventas = ventas.split(',')
arc2.write('Total:' + str(total))
```

Programación—Certamen 3 - 5 de Julio de 2017

Nombre

Rol — Paralelo

2. [**40 %**] Un concesionario de una autopista mantiene información relevante de sus clientes en 3 archivos: Un registro diario de patentes que circulan por ella, la información de las patentes que contienen o no contienen TAG (indicado con un 1 o un 0 respectivamente) y por último, la información de los dueños y de sus patentes asociadas.
A modo de ejemplo se presenta cada uno de los archivos en el cuadro a continuación:

`registro_diario.txt`

```
abmn32
crtj12
dflp11
hb5101
```

`info_duenos.txt`

```
Alex Perez;crtj12,hb5101,kcfl36,emda16
Aquiles Castro ;abmn32,tljg99,avrv33
Maria Gana;ab7677
Fede Santos;utfs90,dflp11
```

De la información de ejemplo se desprende que Alex Perez tiene cuatro vehículos registrados en la autopista con patentes `crtj12`, `hb5101`, `kcf136` y `emda16`. De estos solo `crtj12` cuenta con dispositivo TAG y los otros tres no. Además, los vehículos con patente `crtj12` y `hb5101` pasaron durante el día por la autopista.
Siguiendo con el ejemplo, Maria Gana tiene solamente un vehículo y éste no circuló durante el día por la autopista.
Considere que en el archivo `info_duenos` cada persona aparece solamente una vez y que puede tener un número variable de patentes.

`patentes.txt`

```
crtj12,1
abmn32,0
hb5101,0
dflp11,1
tljg99,0
jfzo10,0
kcfl36,0
utfs90,0
ab7677,0
avrv33,0
emda16,0
```

a) Se necesita saber cuantos vehículos tienen cada usuario. Para esto contruya una función `patentes_por_dueno(archivo_dueños)` que recibe el nombre del archivo que contiene la información de los dueños y retorna un diccionario. Este diccionario debe tener como llave el nombre y apellido de la persona y como valor el número de vehículos que posee.

```
>>> patentes_por_dueno('info_duenos.txt')
{'Alex Perez': 4, 'Aquiles Castro ': 3, 'Fede Santos': 2,
'Maria Gana': 1}
```

b) Se requiere listar a aquellas patentes que circularon durante el día por la autopista y que no disponen de TAG. Para esto, construya la función `patentes_multadas(archivo_registro, archivo_patentes)` que recibe los nombres de los archivos que contienen el registro diario de patentes y el registro general de patentes respectivamente. Esta función debe retornar una lista con las patentes que satisfacen el criterio ya mencionado.

```
>>> patentes_multadas('registro_diario.txt', 'patentes.txt')
['abmn32', 'hb5101']
```

c) Se necesita listar a su(s) respectivo(s) dueño(s). Para esto construya la función `personas_multadas(archivo_registro, archivo_patentes, archivo_dueños)` que recibe los nombres de los archivos que contienen el registro diario de patentes, el registro general de patentes y la información de los dueños de automóviles respectivamente. Esta función debe retornar un listado con los nombres de los dueños de las patentes multadas. Este listado no debe contener nombres repetidos.

```
>>> personas_multadas('registro_diario.txt', 'patentes.txt', '
   info_duenos.txt')
['Aquiles Castro ', 'Alex Perez']
```

Programación—Certamen 3 - 5 de Julio de 2017

Nombre

Rol — Paralelo

3. **[40 %]** Una pequeña empresa chocolatera tiene una base de datos nutricional de materia primas que usa para crear su mercancía. Los archivos de materia prima tienen en la primera línea el formato: `nombre producto, marca producto, cantidad en gramos`. Luego, desde la segunda línea se indica la información nutricional del producto; excepto por las calorías, todos los valores están en gramos. Tener en cuenta, que el proceso de medición de cada materia prima se encarga de guardar los datos en los archivos con los valores de cada porción en **gramos**, sin importar si son líquidos o sólidos.

CHOCOLATEAMARGO.txt

```
CHOCOLATE AMARGO,LA FETE,100
AZUCARES 10
CALORIAS 400
CARBOHIDRATOS 40
GRASAS 20
PROTEINAS 28
```

LICORMENTA.txt

```
LICOR DE MENTA,LIDER,4
AZUCARES 1.6
CALORIAS 20
CARBOHIDRATOS 1.6
GRASAS 0
PROTEINAS 0.2
```

La receta de un producto tiene la estructura indicada en `RECETA:CHOCOLATE_CON_MENTA.txt`. La primera línea indica su peso en gramos, mientras que la segunda línea en adelante se describen las cantidades en gramos de los ingredientes a utilizar.

RECETA:CHOCOLATE_CON_MENTA.txt

```
160
CHOCOLATEAMARGO 150
LICORMENTA 10
```

Con estos archivos, se puede obtener la información nutricional aproximada de los productos. Se le pide a usted que escriba la función `calcular_total(archivo_receta)` que genere un archivo con la información nutricional del nuevo producto. El nombre de este nuevo archivo debe ser igual al nombre que tiene la receta, pero con el prefijo: `NUTRICION`. Para el ejemplo de `RECETA:CHOCOLATE_CON_MENTA.txt`, el nuevo archivo debe ser `NUTRICION:CHOCOLATE_CON_MENTA.txt`.

Tenga en cuenta que el cálculo de la infomación nutricional de la receta debe considerar la información nutricional de cada materia prima (o ingrediente), la cual debe debe ser proporcional a la cantidad de gramos indicada en la receta. Un ejemplo para la función para `RECETA:CHOCOLATE_CON_MENTA.txt` es:

NUTRICION:CHOCOLATE_CON_MENTA.txt

```
CHOCOLATE CON MENTA,160.0
**********
AZUCARES 19.0
CALORIAS 650.0
CARBOHIDRATOS 64.0
GRASAS 30.0
PROTEINAS 42.5
```

Tenga en cuenta que el orden de la información nutricional está ordenada alfabéticamente.