Karrito

Una empresa vende productos e insumos de informática desde su sitio en Internet. Los datos de las compras de los usuarios se mantienen en carritos individuales (un carrito es una dupla `(nombre comprador,[(producto,cantidad)])`. Los carritos se consolidan en una lista:

```
carroSanti  =("Santiago", [("WD TV Live", 1),
             ("Cable HDMI 1.3", 3)])
carroMatias =("Matias",[("Torre DVD", 3),
             ("Cartucho Epson D40",1),
             ("Papel adhesivo DVD", 120)])
carroSilvia =("Silvia",[("Monitor DELL WF424",1),
             ("Teclado 101",1),
             ("Mouse MX500",1)])
carroFede =  ("Fede", [("Cable HDMI 1.3", 1),
             ("Teclado 101",3)] )

compras = [ carroSanti, carroMatias, carroSilvia, carroFede]
```

Tenemos además la lista precio de productos:

```
productos = [("HP P1005", 526.5),("Epson C420", 415),
             ("Toner HP 22 Black",  153.12),
             ("Monitor DELL WF424",1322),
             ("WD TV Live", 745.3),
             ("Gabinete Vitsuba G240",102.94),
             ("Cartucho Epson D40", 52.49),
             ("Papel adhesivo DVD",0.49),
             ("Cable HDMI 1.3",40.99),
             ("Torre DVD", 19.99),
             ("Teclado 101",19.50),
             ("Mouse MX500",79)]
```

Codificar las funciones indicadas a continuación. En la resolución tenga presente que deben aparecer aplicados, al menos una vez, los siguientes conceptos:

- Aplicación Parcial
- Composición
- Funciones de Orden superior
- Listas por comprensión

No se puede usar recursividad, a menos que esté indicado en el punto.

1) comproProducto/2, que recibe un nombre de producto y un carrito, y retorna True si el carrito contiene el producto:

```
>comproProducto "Torre DVD" carroFede
False
```

2) quienesCompraron/1, que recibe un nombre de producto y retorna los nombres de las personas que compraron ese producto:

```
>quienesCompraron "Teclado 101"
["Silvia","Fede"]
```

3) cuantoCompro/2, que recibe un nombre de producto y una dupla carrito, y retorna la cantidad de ítems comprados, o cero si no compró nada

```
>cuantoCompro "Teclado 101" carroSanti
0
>cuantoCompro "Teclado 101" carroFede
3
```

4) cantidadVendida/1, que recibe un nombre de producto y retorna el total vendido de ese producto:

```
>cantidadVendida "Teclado 101"
4
>cantidadVendida "HP P1005"
0
```

5)

a. queCompro/1, que recibe un nombre de usuario y retorna el carrito de esa persona:

```
>queCompro "Fede"
[("Cable HDMI 1.3", 1),("Teclado 101",3)]
```

b. precioDe/1, que recibe un nombre de producto y retorna el precio unitario de ese producto:

```
>precioDe "WD TV Live"
745.3
```

c. gastoDeUsuario/1, que recibe un nombre de usuario y retorna el gasto total de ese usuario:

```
>gastoDeUsuario "Santiago"
868.27
```

d. mayorGastador, que retorna el nombre del usuario que gastó más.

```
>mayorGastador
"Silvia"
```

En este punto se puede usar recursividad.

6) Como queremos segmentar los compradores de distinta manera, definimos esta función:

```
segmentar criterio = (map nombreComprador . filter criterio) compras
```

Utilice esta función para determinar:

- Qué clientes compraron más de 2 productos diferentes (ejemplo: Matías que llevó 3 productos)
- Qué clientes compraron más de 2 teclados 101 (ejemplo: Fede)
- Qué clientes tienen nombres de más de 5 letras (ejemplo: Santiago)

Nota: en este punto debe definir la función criterio sólo con composición de funciones (no puede usar funciones auxiliares ni expresiones lambda ni definiciones locales)

7) Defina el tipo de la siguiente función

```
funcionLoca a b c d | a > b     = (c . d) b
                    | otherwise = (c . d) a
```