Práctica 1b
Salvador Martín Barcia
Borja Pérez Bernardos

Dudas:

- Las operaciones con 2 argumentos deben ser operadores en lugar de funciones.

a)

- Para representar los artículos, las ventas y las facturas hemos utilizado type.
- Las funciones que hemos implementado como operadores son: fusión de 2 facturas, precio total de un artículo en una factura, búsqueda en una factura de las ventas relativas a un artículo, búsqueda en una factura de las ventas relativas a una lista de artículos, eliminación en una factura de las ventas relativas a un determinado artículo y eliminación en una factura de las ventas de aquellas relativas a una cantidad menor que una determinada.

b)

- Para representar los tipos hemos usado data. En el caso del **Articulo** y de la **Factura** usamos un único constructor mientras que en **Venta** usamos uno para **VentaUnitaria**, otro para **Venta** y un último para el operador constructor **(:+)**.
- Al cambiar la declaración de los tipos hemos tenido que cambiar todas las funciones adaptándolas al nuevo formato.

c)

- Hemos cambiado ***precioFactura*** (precio de una factura) para que sea de orden superior y reciba la función ***precioFactura*****ux**.
- Hemos cambiado ***precArtFact*** (precio total de un artículo en una factura) para que sea de orden superior y reciba como uno de sus argumentos ***precArtFactaux*****.**
- Hemos cambiado ***busquedaDeVentas1*** (búsqueda en una factura de las ventas relativas a un artículo) para que sea una lista por comprensión ya que reduce bastante las líneas de código.
- Hemos cambiado ***busquedaDeVentas2*** (búsqueda en una factura de las ventas relativas a una lista de artículos) para que sea de orden superior.
- Hemos cambiado ***elim1*** (eliminación en una factura de las ventas relativas a un determinado artículo) para que sea una lista por comprensión.
- Hemos cambiado ***elim2*** (eliminación en una factura de las ventas de aquellas relativas a una cantidad menor que una determinada) para que sea una lista por comprensión.

d)

- Hemos cambiado las funciones ***cantidadVenta*** y ***precioVenta*** para que cuando reciban precios o cantidades negativas, se lance un error.

- Hemos creado la función ***compruebaFactura*** que comprobará si hay artículos en una factura que contengan el mismo id pero diferentes precios/nombres. Esta función devolverá True si en la factura los artículos que tienen mismo id, tienen los mismos precios/nombres, y devolverá False en caso contrario. Esta función la utilizaremos en el resto de funciones que reciben alguna factura para comprobar que la factura es correcta.
- Como la función ***error*** para la ejecución del código, en este apartado no hemos podido ejecutar todos los tests a la vez. Hemos tenido que ejecutarlos uno a uno y hemos puesto el resultado al final del fichero. Es por esto que hemos comentado todos los print del main del **ApartadoD**.

e)

- Hicimos modificaciones en las definiciones de Factura, Venta y Articulo para que fueran GADTs y poder utilizar los tipos de las clases Fractional e Integral. El precio del tipo Articulo pasó a ser de un tipo de la clase Fractional y el id de Articulo y la cantidad de Venta a ser un tipo de la clase Integral.
- Esto hizo que tuviéramos que modificar el resto de funciones para que fueran polimórficas de tipos de las clases Fractional e Integral.

f)(*)

- Hemos nombrado la función precio2 en vez de precio porque la función precio ya existía como getter de la definición del tipo Articulo.
- Hemos utilizado las instancias para definir la función precio2 dependiendo del tipo del argumento que recibe, es decir, si recibe una Factura, que ejecute la función precioFactura y si recibe una Venta, ejecute precioVenta.

g)

```
data Arbol a = Arb a [Arbol a]
data Factura = Factura {ventas :: Arbol Venta}
```

- Hemos cambiado la estructura de Factura quitando la lista de ventas y convirtiéndolo en un Arbol de ventas.
- El tipo Arbol tiene como segundo parámetro una lista ya que cuando esta es vacía, significa que es el último nodo de la estructura Arbol.
- Como el tipo Factura ha cambiado, el resto de funciones las hemos modificado para adaptarlas al nuevo cambio.

h)

- Decidimos hacer un constructor para cada tipo de Venta teniendo VentaADomicilio, Venta y VentaPrecioMayor como constructores de tipo Venta. Los elementos de tipo Venta siguen teniendo como parámetros un artículo y cantidad de este.
- Para el tipo Articulo hemos añadido los constructores Alimento y Electrodomestico para añadir diferentes tipos de Articulo.

- Hemos añadido el parámetro “promo” e “impuesto” a los constructores de Articulo, donde el parámetro promo es de tipo Promocion que hemos creado para expresar los diferentes tipos de promociones. El parámetro impuesto lo hemos expresado de tipo Float ya que facilita luego los cálculos.
- Hemos creado la función getpromocion para poder obtener un valor numérico de cada tipo de Promocion.
- En la función precioVenta cuando recibe el tipo VentaPrecioMayor añadimos un extra que también se podría hacer con un porcentaje extra, para que este sea mayor que la Venta normal. También cuando recibe una Venta de tipo VentaADomicilio, añadimos 5 euros como coste de transporte al domicilio.
- Hemos modificado las instancias de Show para acomodarlas a los nuevos tipos, el resto de funciones no han variado mucho.