Una empresa de servicios varios ofrece la posibilidad de realizar compras varias desde un sitio en Internet. Los usuarios solo deben registrarse e ir eligiendo cada artículo, como también su cantidad.
Este sistema corre un proceso al finalizar el día que genera la siguiente base de conocimientos:

```
precioUnitario(producto(tomate), 12.50).
precioUnitario(producto(leche, sancor), 2.45).
precioUnitario(producto(papa), 4.50).
precioUnitario(producto(yogur, laSerenisima), 1.75).
precioUnitario(producto(yogur, sancor), 1.65).
precioUnitario(producto(yogur, manfrey), 1.15).
precioUnitario(producto(fósforos, los3Patitos), 1).
```

El producto se representa como un functor con:

- El nombre de un producto genérico, o bien
- El nombre de un producto y la marca que lo comercializa

```
compro(leo, producto(tomate), 2).
compro(leo, producto(yogur, manfrey), 10).
compro(leo, producto(papa), 1).
compro(nico, producto(tomate), 3).
compro(flor, producto(yogur, laSerenisima), 2).
compro(flor, producto(leche, sancor), 4).

marcaImportante(sancor).
marcaImportante(laSerenisima).
```

Se pide:

1) Realice las consultas que permitan determinar
   a. quiénes compraron productos de Sancor (debe devolver los individuos leo y flor)
   b. qué compró Leo
   c. si Leo compró 2 cosas de algún producto (debe decirme que sí).

2) Resuelva el predicado **cuantoGasto/2** que relaciona una persona con el total que gastó.

? cuantoGasto(flor, Total)
Total = 13.5 (3.5 de los dos yogures y 10 de las cuatro leches)

   a. Indicar si el predicado **cuantoGasto/2** es inversible. Justificar
   b. ¿Qué otros conceptos aparecen? ¿por qué?

3) Codifique el predicado **esMarquero/1**, que relaciona las personas que sólo compran cosas de marcas importantes. Pensarlo:
   a. Que acepte a las personas que compran productos genéricos (como el tomate, la papa, etc.)
      En ese caso la consulta esMarquero(Quien) incluye a nico y a flor.
   b. Que no acepte a las personas que compran productos genéricos.
      En ese caso la consulta esMarquero(Quien) sólo incluye a flor.

   El predicado a generar debe ser inversible.

4) Resuelva el predicado **esGastador/1**, donde una persona es gastadora si:
   - Compró más de 5 productos, o
   - Compró un producto de La Serenísima (como Flor), o
   - El total de la compra excede los 100$.

   El predicado a generar debe ser inversible.

5) Genere el predicado **ganador/2** que indica si la cantidad total de ventas de un producto es mayor que la cantidad total de ventas de otro producto.

   Ejemplos:
   ?- ganador(producto(leche, sancor), X).
   X = producto(papa) ; → *4 de leche sancor vs. 1 de papa*
   X = producto(yogur, laSerenisima) ; → *4 de leche sancor vs. 2 de yogur*

   ?- ganador(producto(papa), producto(yogur, laSerenisima)).
   No

   ?- ganador(producto(yogur, laSerenisima), producto(papa)).
   Yes

   Indique qué consecuencias (ventajas o desventajas) trajo trabajar con distintos tipos de datos para representar los productos de marca y los genéricos.

6) Agregamos ahora en la base de conocimientos los siguientes hechos:

   amigo(leo, flor).
   amigo(flor, nico).

   Resuelva el predicado **puedePrestar/3**, que indica si una persona A puede prestarle a otra persona B un producto, para ello:

   - La persona A tiene que haber comprado ese producto (y la persona B no)
   - O bien, algún amigo de A puede prestarle a B ese producto.

   a. Indicar qué necesita el predicado para ser inversible. Justificar.
   b. Indique dónde aparece el concepto de recursividad. Justificar.

Notas:

- Se puede utilizar el predicado **sum/2**, **length/2** y **sinRepetidos/2**, asumiéndose como predicados que vienen con el lenguaje.
- Revisar lo que se pide y en caso de duda consulte a su docente.

NOMBRE:	
NOTA:	

Soluciones:
Punto 1)

```
?- compro(Quien, producto(_, sancor), _).
Quien = leo ;
Quien = flor ;
No

?- compro(leo, X, _).
X = producto(tomate) ;
X = producto(yogur, sancor) ;
X = producto(papa) ;
No
(también vale generar una incógnita a la variable anónima)

?- compro(leo, _, 2).
Yes
```

2) persona(Persona):-compro(Persona, _, _).

```
montoCompra(Persona, Total):-
    compro(Persona, Producto, Cantidad),
    precioUnitario(Producto, Precio),
    Total is Precio * Cantidad.

cuantoGasto(Persona, Monto):-persona(Persona),
   findall(Total, montoCompra(Persona, Total), Totales),
   sum(Totales, Monto).
```

Predicado generador, que puedo definir "a mano" o generando soluciones repetidas… (mucho no me importa que haya repetidos) ¿Para qué? Para ligar Persona al utilizar el findall, y como corolario hago inversible cuantoGasto

Menos expresivo, lo podríamos meter en un solo predicado:

```
cuantoGasto3(Persona, Monto):-persona(Persona),
   findall(Total, (compro(Persona, Producto, Cantidad),
                precioUnitario(Producto, Precio),
                Total is Precio * Cantidad), Totales),
   suma(Totales, Monto).
```

Otra opción (la que seguro la mayoría va a usar):

```
sumar(_, [], 0).
sumar(Persona, [Producto|Ps], Total):-sumar(Persona, Ps, TotalPs),
   precioUnitario(Producto, Precio),
   compro(Persona, Producto, Cantidad),
   Total is (Precio * Cantidad) + TotalPs.

cuantoGasto2(Persona, Monto):-persona(Persona),
   findall(Producto, compro(Persona, Producto, _), Productos),
   sumar(Persona, Productos, Monto).
```

Al contestar acerca de la inversibilidad, está bueno que lo hagan pensando en el tipo de consultas que yo puedo hacer (unificando o no variables a individuos):

```
cuantoGasto(Persona, Monto),
cuantoGasto(leo, Monto),
cuantoGasto(Persona, 100) .
```

Son distintas consultas, cuánto gastaron todos, cuánto gastó leo o quién gasto 100 $.

3) a)

```
esMarquero(Persona):-persona(Persona),
   forall(compro(Persona, producto(_, Marca), _), marcaImportante(Marca)).
```

b)

```
esMarquero2(Persona):-persona(Persona),
   not(compro(Persona, producto(_), _)),
   forall(compro(Persona, producto(_, Marca), _), marcaImportante(Marca)).
```

El predicado esMarquero es inversible si yo unifico Persona antes de utilizar el forall. De otra manera lo que buscará es que todas las personas hayan comprado productos de marcas importantes.
Lo que logro con un predicado generador es preguntar persona por persona si cumple ser "marquera".

4)

```
esGastador(Persona):-persona(Persona),
   findall(Producto, compro(Persona, Producto, _), Productos),
   length(Productos, Cantidad), Cantidad > 5.
esGastador(Persona):-compro(Persona, producto(_,laSerenisima), _).
esGastador(Persona):-cuantoGasto(Persona, Total), Total > 100.
```

5)

```
producto(Producto):-compro(_, Producto, _).

totalPorProducto(Producto, Monto):-producto(Producto),
    findall(Total, compro(_, Producto, Total), Totales),
    suma(Totales, Monto).

ganador(Producto1, Producto2):-totalPorProducto(Producto1, Total1),
    totalPorProducto(Producto2, Total2),
    Total1 > Total2.
```

Trabajar con functores me facilita comparar los productos entre sí. De otra manera tendría que trabajar las cláusulas por separado, comparando:

- Productos genéricos vs. productos genéricos
- Productos de marca vs. productos genéricos
- Productos de marca vs. productos de marca
- Productos genéricos vs. productos de marca

Ejemplo sin usar functores:

Tendría dos cláusulas compro:

```
compro(nico, tomate, 3).
compro(flor, yogur, laSerenisima, 2).

totalPorProducto(Producto, Monto):-producto(Producto),
    findall(Total, compro(_, Producto, Total), Totales),
    suma(Totales, Monto).

totalPorProducto(Producto, Marca, Monto):-producto(Producto),
    findall(Total, compro(_, Producto, Marca, Total), Totales),
    suma(Totales, Monto).
```

Y lo mismo habría que hacer al comparar dos productos, para ver si cada uno es de marca o genérico.

O sea, tomando la descripción y buscando sobre un predicado precioUnitario de distinta aridad se me complica más. De esta manera yo trabajo sobre el concepto de productos, independientemente de si son de marca o no.

Las consultas del enunciado tenían la intención de ayudar en la respuesta.

6)

```
puedePrestar(Persona1, Persona2, Producto):-persona(Persona1),
    persona(Persona2),
    Persona1 \= Persona2,
    compro(Persona1, Producto, _),
    not(compro(Persona2, Producto, _)).

puedePrestar(Persona1, Persona2, Producto):-persona(Persona1),
    persona(Persona2),
    Persona1 \= Persona2,
    amigo(Persona1, Persona3),
    puedePrestar(Persona3, Persona2, Producto).
```

Aquí aparece la recursividad: yo predicado estoy relacionándome conmigo mismo

Lo que necesitaba el predicado para ser inversible es que la Persona2 esté unificada al preguntar por not(compro(Persona2, Producto, _), ese es el único predicado generador necesario. Los otros predicados generadores sólo son por reglas de negocio (Persona1 debe ser distinta de Persona2).

En la segunda cláusula no es obligatorio utilizar predicados generadores, lo que pasa es que nos sirve para evitar consultas erróneas, como puedePrestar(flor, flor, QueCosa).

A los fines prácticos de corrección importa que esté bien definida la llamada a la cláusula recursiva.

Punto	Qué evaluamos
1	Variables vs. individuos. Consultas individuales vs. existenciales. Uso de variable anónima y de functores.
2	Predicados de orden superior (findall). Generación. Inversibilidad.
3	Predicados de orden superior (forall). Generación. Negación. Declaratividad (ausencia de algoritmo).
4	Definición de consultas con or. Predicados de orden superior. Expresividad.
5	Predicados de orden superior. Functores. +… …el foco en los puntos anteriores había estado en las cosas que compró X. Acá vemos que armando una base de conocimientos con hechos individuales puedo consultar cuánto vendí de un producto, cosa que en funcional requiere “dar vuelta” una lista.
6	Recursividad. Generación. Inversibilidad