1) Para obtener la reversa de una lista se programaron dos soluciones diferentes:

Pseudocódigo

```
type
   LISTA = array[1..longitudMaxima] of integer;
   FUNCION = integer -> integer;
proc filter(lis: LISTA, f: FUNCION, var lis2: LISTA)
begin
        j ← 1;
        recorrer i de 1 a longitudMaxima
        begin
                if f(lis[i]) then
                begin
                    lis2[j] = lis[i];
                    j ← j + 1;
                end
        end
end
```

Haskell

```
filter f lis = [e | e <-lis, f e]
```

a) El pseudocódigo de la solución de la izquierda incluye una característica que en la materia vimos como asociada a la programación funcional. Indicar cuál es esta característica, y marcar en el pseudocódigo dónde aparece justificando.

b) Haga una comparación de ambas soluciones en términos de:

- Declaratividad: qué solución tiende a ser más declarativa y por qué.
- La necesidad de definir los tipos en una y otra solución, relacione con algún concepto visto en la materia.

c) Ahora se pide conocer los primeros n elementos de la lista filtrada. Las nuevas soluciones son:

Pseudocódigo

```
type
   LISTA = array[1..longitudMaxima] of integer;
   FUNCION = integer -> integer;
proc filterN(lis: LISTA, f: FUNCION,
            n: integer, var lis2: LISTA)
begin
        j ← 1;
        recorrer i de 1 a longitudMaxima
        begin
                if f(lis[i]) then
                begin
                    lis2[j] = lis[i];
                    if f(j == n) then
                        break;
                    else
                        j ← j + 1;
                    end
                end
        end
end
```

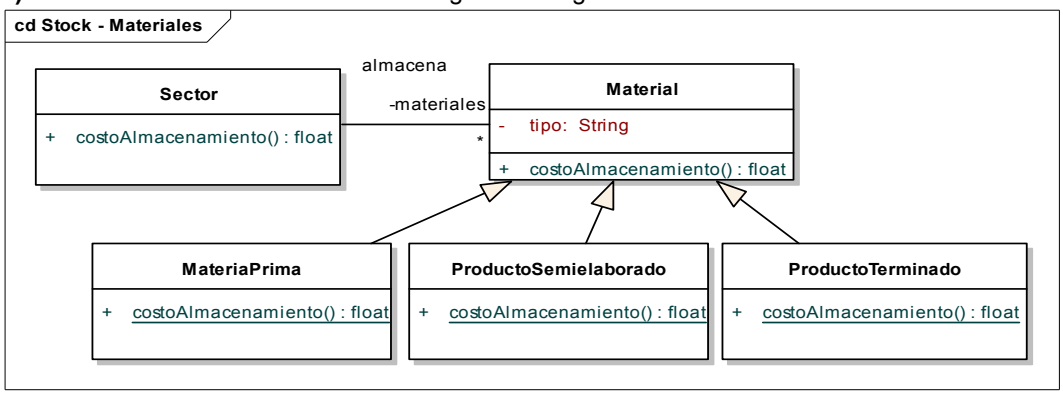
Haskell

```
take 0 _ = []
take n (x:xs) = x:take (n-1) xs

Desde una pantalla de consulta,
supongamos que n es 3:
> (take 3 . filter even) [1, 8, 3, 2, 52, 74]
```

1) Independientemente de la eficiencia, ¿qué ventaja le ve a la versión en Haskell? ¿Qué conceptos ayudan (marcarlos en el código y justificar en qué ayuda cada concepto)?
2) ¿Qué tipo de análisis puedo hacer con la versión en pseudocódigo que no puedo hacer con la versión en Haskell?
3) Es posible desarrollar una función take en pseudocódigo que imite la solución haskellera. Indique qué cambios habría que hacer (no hace falta codificar, sólo indicar qué modificaciones habría que hacer en el código). ¿Qué concepto/s está/n presente/s al hacer este cambio?
4) ¿Es posible trabajar con una lista potencialmente infinita en cada solución? Justifique. Para el caso del pseudocódigo, considerar por separado la versión presentada y la que ud. propuso en el punto anterior.

2) En un sistema de Stock tenemos el siguiente diagrama de clases:



#Material
costoAlmacenamiento
"Determinar el costo de almacenamiento"
(tipo = "P") "Materia prima"
ifTrue: [MateriaPrima costoAlmacenamiento: self]
(tipo = "S") "Semielaborado"
ifTrue: [ProductoSemiElaborado costoAlmacenamiento: self]
(tipo = "T") "Terminado"
ifTrue: [ProductoTerminado costoAlmacenamiento: self]

En MateriaPrima, ProductoSemiElaborado y ProductoTerminado hay un método de clase que calcula el costo de almacenamiento (es distinto para cada uno).

En la clase Sector tenemos el siguiente código:
costoAlmacenamiento
"Determino el costo de almacenamiento de cada sector"
^materiales inject: 0 into: [:acum :material | acum + material costoAlmacenamiento]

a) ¿Qué correcciones haría en el código y por qué? (No es necesario que codifique, solamente indique los cambios que haría y justifique qué conceptos del paradigma está tratando de aprovechar)
b) ¿Qué impacto sufre por estos cambios la clase Sector? Justifique su respuesta relacionándolo con algún concepto visto en la cursada.
c) Si en el sector queremos guardar encomiendas, que no participan del proceso de fabricación (no tiene sentido considerarlo un material). El costo de almacenamiento lo calcula el encargado del depósito (para el sistema es un monto fijo distinto para cada encomienda) ¿Dónde ubicaría la clase Encomienda en el diseño de la nueva solución?
d) ¿Qué impacto sufre por estos cambios la clase Sector? Justifique su respuesta relacionándolo con algún concepto visto en la cursada.

3) Para una aplicación que administra los surtidores de una estación de servicio (a desarrollar en PROLOG) se han propuesto dos opciones para modelar la carga de los surtidores:

Modelarlo como una lista de functores	**Modelarlo con cláusulas independientes**
infoSurtidor([surtidor("RVM363", "Nafta 5000", 25, 72.10, fecha(22, 02, 2009)), surtidor("BWF003", "Nafta 8000", 20, 86.10, fecha(22, 02, 2009)), …]). …	surtidor("RVM363", "Nafta 5000", 25, 72.10, fecha(22, 02, 2009)). surtidor("BWF003", "Nafta 8000", 20, 86.10, fecha(22, 02, 2009)). …

En ambos casos, la información se agrega en el orden en que los autos son atendidos, se genera el programa PROLOG, y se usa para obtener información.

a)
De cada ítem que viene a continuación, indicar si se puede hacer usando el modelo de lista de functores, se puede hacer usando el modelo de cláusulas independientes, se puede hacer con cualquiera de los dos, o no se puede hacer con ninguno de los dos.

1. Implementar un predicado que relacione una fecha con el total de nafta despachada en esa fecha.
2. Implementar un predicado fueAtendidoAntes(Auto1,Auto2,Fecha) que se verifique si ambos autos fueron atendidos en la fecha indicada, y además Auto1 fue atendido antes que Auto2.
3. Consultar, usando en la consulta únicamente el predicado que define el modelo, qué autos cargaron la nafta de 5000 octanos.

b) En cualquiera de los dos modelos, ¿cómo representaría la información de un cliente "RTU 291" que se arrepintió y no cargó nafta? ¿Con qué concepto está relacionado y por qué?

c) Trabajando con el modelo de cláusulas independientes, agregamos la siguiente cláusula:
auto(Auto, Monto):-surtidor(Auto, _, _, Monto, _).

¿Es posible saber:

- Todos los autos que cargaron 25 pesos
- Todos los importes que cargó el auto "RVM363"
- El importe que cargó cada auto
- Si el auto "RVM363" cargó 26 pesos alguna vez

sin necesidad de cambiar la definición del predicado auto? Justifique por qué, indique con qué concepto está relacionado y dónde aparece.