

Enunciado general

TRABAJO PRÁCTICO – PARADIGMA FUNCIONAL

La secretaría de agricultura quiere hacer una aplicación que maneje las características de las manzanas que se producen en la Argentina.

Para eso se le va a poner a cada cajón un código de barras con: cantidad de manzanas, peso de la más chica, peso de la más grande. Esa información se va a recolectar agrupada por partidas (una partida es un montón de cajones que se recolectan en la misma localidad), y luego las partidas se agrupan por provincias.

En una prueba de concepto de la aplicación se usa esta información, donde se describen cinco partidas y una provincia:

```
partidaAreco = [(28,204,293), (31,198,244), (26,184,314)]
partidaLujan = [(24,243,294), (34,201,262), (27,204,288), (9,200,209)]
partidaMercedes = [(28,243,301), (26,232,289), (28,250,308)]
partidaRojas = [(19,223,242), (21,224,243), (22,231,303)]
partidaMoreno = [(19,224,236), (21,224,242)]

partidasBsAs = [partidaAreco, partidaLujan, partidaMercedes,
                 partidaRojas, partidaMoreno]
```

Se proveen estas funciones:

```
fst3 (x,y,z) = x
snd3 (x,y,z) = y
trd3 (x,y,z) = z
divide n m = mod n m == 0
primo n = all (not . divide n) [2..round (sqrt (fromInteger n))]
```

1° Entrega

Fecha límite de entrega: martes 13/04 22:00 hs.

Conceptos a evaluar:

- Recursividad
- Listas por comprensión

1. **pesosMinMax/1**, recibe una partida y devuelve una lista de pares (peso mínimo cajón, peso máximo cajón) para cada cajón de la partida.
El peso mínimo de un cajón es cantidad * peso de la manzana más chica, el peso máximo es cantidad * peso de la manzana más grande

P.ej. la consulta

```
> pesosMinMax partidaRojas
```

devuelve [(4237, 4598), (4704, 5103), (5082, 6666)], donde 4237 es 19*223, 4598 es 19*242, etc.

Resolverlo aplicando

- a) recursividad
- b) listas por comprensión

2. **partidaCreciente1**, recibe una partida e indica si el número de manzanas de cada cajón es creciente, o sea, el número para cada cajón es menor al que le sigue.
En el ejemplo, las partidas de Rojas (19 < 21 < 22) y la de Moreno (19 < 21) son crecientes, el resto no (p.ej. la de Areco no porque 31 > 26).

Qué se espera en cada entrega

- Un archivo .hs con el desarrollo de cada función (preferentemente, anotar con un comentario previo el punto que se está resolviendo). Ej:

```
-- Punto 1  
pesosMinMax ...
```

- Otro archivo que documente las pruebas que se hicieron (las consultas al motor Haskell)

Importante: los conceptos que se evalúan en cada entrega deben estar en el desarrollo de las funciones (al menos una vez).

Cómo se entrega

Se debe mandar un mail al docente corrector. Se considera cumplida la entrega con el ok del docente.

2° Entrega

Fecha límite de entrega: martes 20/04 22:00 hs.

Conceptos a evaluar:

- Composición de funciones
- Funciones de orden superior

3. **deExportacion/1**, recibe una partida e indica si es o no de exportación. Para que una partida sea de exportación deben cumplirse estas tres condiciones:
- tener más de 70 manzanas
 - que todas las manzanas alcancen los 200 gramos
 - que al menos una manzana alcance los 300 gramos.

En el ejemplo, la única partida de exportación es la de Mercedes, porque: en la de Areco no todas las manzanas alcanzan 200 gramos (hay al menos una de 198), en la de Luján ninguna manzana alcanza 300 gramos, ni la de Rojas ni la de Moreno llegan a 70 manzanas.

Importante: armar funciones auxiliares para cada condición, van a ser útiles más adelante.

4.

- 4.a. **between/3**, recibe tres números e indica si el tercero está entre los dos primeros.

Consultas de ejemplo

```
> between 3 5 4      -- True
> between 3 5 5      -- True
> between 3 5 6      -- False
```

- 4.b. **cuantasPartidasVerifican/2**, recibe una condición y una provincia, e indica cuántas partidas de la provincia verifican la condición.

- 4.c. Indicar las consultas que devuelven, para la info de la provincia de Buenos Aires y usando `cuantasPartidasVerifican`, lo siguiente:

- cuántas partidas de exportación hay
- cuántas partidas hay en las que todas las manzanas alcanzan al menos 210 gramos.
- cuántas partidas hay que tienen exactamente 3 cajones.
- cuántas partidas hay con entre 60 y 80 manzanas.

3° Entrega

Fecha límite de entrega: martes 27/04 22:00 hs.

Conceptos a evaluar:

- Aplicación parcial
- Expresiones lambda
- Funciones de orden superior
- Composición
- Inferencia de tipos

5.

- 5.a. **partidaBizarra**/1, recibe una partida y devuelve True si tiene al menos dos cajones para los que el peso de la manzana más grande + 4 no es primo; y False en caso contrario.
Este hay que hacerlo en una sola línea, o sea sin funciones auxiliares, y tampoco se pueden usar listas por comprensión.

P.ej. la consulta

```
partidaBizarra partidaMercedes  
devuelve True
```

1º cajón: $301 + 4 = 305$ (no es primo, 1º que cumple)

2º cajón: $289 + 4 = 293$ (es primo)

3º cajón: $308 + 4 = 312$ (no es primo, 2º que cumple)

- 5.b. **sumaFns**/2, que recibe una lista de funciones y un valor, y devuelve la suma del resultado de aplicarle todas las funciones al valor.

P.ej. la consulta

```
sumaFns [(3*), (min 4), (2+)] 5  
devuelve 26:  $15 + 4 + 7$ .
```

Resolverlo con foldl + expresiones lambda

- 5.c. **cuentaBizarra**/1, recibe una partida y devuelve, para cada cajón, la suma del doble del peso de la manzana más chica más el peso de la manzana más grande.

P.ej. la consulta

```
cuentaBizarra partidaAreco  
devuelve [701, 640, 682]:, 701 es  $(204 * 2) + 293$ . Usar sumaFns.
```

6.

- 6.a. **cajonesYManzanas**/1, recibe una provincia y devuelve una lista de pares (cantidad de cajones, cantidad total de manzanas) para cada partida de la provincia en la que todas las manzanas alcancen al menos 220 gramos. P.ej. la consulta

```
> cajonesYManzanas partidasBsAs  
devuelve [(3, 82), (3, 62), (2, 40)] (partidas de Mercedes, Rojas y Moreno).
```

- 6.b. Dada esta definición de función

```
f1 x y z = x z + y z
```

indicar su tipo, y usarla para obtener la cantidad de cajones más la cantidad de manzanas de la partida de Areco (debería dar 88, 3 cajones y 85 manzanas en total).

Consejo: usar una función longitud definida como

```
longitud = fromInt . length
```