Notas previas

Se pueden usar estas funciones del prelude: all / any / map / filter / elem. También esta función

```
sumList l = foldl (+) 0 l
```

Para aprobar hay que hacer tres de los cinco puntos; y usar al menos tres de estas cinco características

1. recursividad
2. funciones de orden superior
3. composición
4. aplicación parcial
5. listas por comprensión

Obviamente, cuantas más se usen mejor.
De las que se usen, indicar un uso de cada uno en algún lugar aparte de la resolución de los ítems.

Contexto (para los ítems 2 a 5)

Se nos pide desarrollar un programa Haskell que analice las calificaciones obtenidas por los aspirantes a distintas becas y concursos. Cada aspirante rinde cuatro exámenes, de materias a su elección. Los aspirantes se reúnen en planillas según en qué mes rindieron sus exámenes. P.ej.:

```
planillaEnero =
    [("pepe",  [("fisica",6),("algebra",9),("datos",8),("disenio",7)]),
     ("lucia", [("historia",8),("filosofia",9),("algebra",10),("poesia",8)]),
     ("franco",[("algebra",9),("disenio",9),("datos",10),("derecho",8)]),
     ("ana",   [("derecho",7),("algebra",9),("disenio",8),("poesia",9)])
    ]

planillaFebrero =
    [("marta", [("fisica",5),("algebra",6),("datos",7),("disenio",8)]),
     ("pedro", [("historia",8),("derecho",9),("algebra",10),("poesia",8)])
    ]
```

(lo que sigue lo van a necesitar solamente para el ítem 4).
Los requisitos para acceder a una beca o concurso se describen mediante un par (mínima nota total, exámenes que hay que tener), p.ej. el par

```
(35,["algebra","disenio"])
```

representa un requerimiento en el que para cumplirlo la suma de las notas tiene que ser al menos 35, y además hay que haber rendido los exámenes de álgebra y de diseño (no importa qué nota te sacaste en esos exámenes). De los de la planilla de enero, solamente Franco lo cumple; Lucía suma más de 35 pero no tiene Diseño; Pepe tiene las dos pero no llega a 35.

1.

a. Definir la función **maximoF**, que dados una función y una lista, devuelve el elemento de la lista que es máximo para la función. P.ej.

```
maximoF abs [-5..3]                          devuelve -5
maximoF length ["la", "larga", "paz"]        devuelve "larga"
```

Indicar el tipo de la función.

b. Definir la función estaIncluida, que dadas dos listas devuelve True si todos los elementos de la primera son elementos de la segunda, y False en caso contrario. P.ej.

```
estaIncluida [2,4] [1..5]                    devuelve True
estaIncluida [4,6] [1..5]                    devuelve False
```

2.

a. Definir la función **sePresento**, que dados un nombre de alumno y una planilla, indica si los exámenes del alumno están en la planilla. P.ej.

```
sePresento "franco" planillaEnero            devuelve True
sePresento "franco" planillaFebrero          devuelve False
```

¿Cómo preguntarían si Franco se presentó en algún mes (o bien en enero o bien en febrero)? La consulta es `sePresento "franco" ...`, completar cómo sigue.

b. Definir la función **notas**, que que dados un nombre de alumno y una planilla, devuelve la lista de pares con los resultados que obtuvo el alumno. P.ej.

```
notas "franco" planillaEnero
```

devuelve `[("algebra",9),("disenio",9),("datos",10),("derecho",8)]`.

3.

a. Definir la función **mejorTema**, que dados los resultados de un aspirante devuelve la materia en que obtuvo la mejor nota. P.ej.

```
mejorTema [("algebra",9),("datos",10),("derecho",8)]
```

devuelve `"datos"`.
Escribir la consulta que responde el mejor tema de Lucía en la planilla de enero, sin necesidad de repetir los resultados de Lucía en la consulta (ayuda: usar notas además de mejorTema).

b. Definir la función **aproboTodo**, que dados los resultados de un aspirante devuelve True si todas las notas son 7 o más, y False en caso contrario. P.ej.

```
aproboTodo [("algebra",9),("datos",10),("derecho",8)]
aproboTodo [("algebra",9),("datos",5),("derecho",8)]
```

devuelven `True` y `False` respectivamente.

c. Definir la función **rindio**, que dados un nombre de materia y los resultados de un aspirante devuelve True si el aspirante rindió la materia, y False en caso contrario. P.ej.

```
rindio "derecho" [("algebra",9),("datos",10),("derecho",8)]
rindio "filosofia" [("algebra",9),("datos",5),("derecho",8)]
```

devuelven `True` y `False` respectivamente.

4.

a. Definir la función **mejoresTemas**, que dada una planilla devuelve la lista de pares (nombre, materia en la que obtuvo la mejor nota). P.ej.

```
mejoresTemas planillaFebrero
```

devuelve `[("marta","disenio"),("pedro","algebra")]`. Ayuda: usar mejorTema.

b. Definir la función **cumple**, que dados un requerimiento y los resultados de un aspirante indica si con esos resultados se cumple el requerimiento o no. P.ej.

```
cumple (35,["algebra","disenio"]) (notas "lucia" planillaEnero)
cumple (35,["algebra","disenio"]) (notas "franco" planillaEnero)
```

devuelven `False` y `True` respectivamente (ver la explicación en la parte de contexto).

c. Definir la función **alguienCumple**, que dados un requerimiento y una planilla, indica si alguno de los aspirantes incluidos en la planilla cumplen con el requerimiento. P.ej.

```
alguienCumple (35,["algebra","disenio"]) planillaEnero
alguienCumple (35,["algebra","disenio"]) planillaFebrero
```

devuelven `True` y `False` respectivamente. Ayuda: usar cumple.

5.

a. Definir la función **quienesRindieron**, que dados una materia y una planilla, devuelve los nombres de los que rindieron esa materia. P.ej.

```
quienesRindieron "derecho" planillaEnero
```

devuelve `["franco","ana"]`. Ayuda: la función rindio (ítem 3.c) puede ser útil (o no, según cómo se encare la resolución).

b. Definir la funcion **rindieronTodos**, que dados una materia y una planilla, devuelve True si todos los aspirantes de la planilla rindieron la materia, y false en caso contrario. P.ej.

```
rindieronTodos "algebra" planillaEnero
rindieronTodos "poesia" planillaEnero
```

devuelven `True` y `False` respectivamente. Ayuda: otra vez puede venir bien la función rindio.