Notas previas

Se pueden usar estas funciones: elem, map, filter, all, any, sum, y las que se definen a continuación.

```
maximoList [e] = e
maximoList (e:es)
    | e > maximoList es = e
    | otherwise         = maximoList es

minimoList [e] = e
minimoList (e:es)
    | e < minimoList es = e
    | otherwise         = minimoList es
```

Algunos ítems indican pautas para su resolución, una resolución que no siga esas pautas se considerará inválida aunque "ande".

Contexto

Se nos pide desarrollar un programa Haskell que analice las calificaciones obtenidas por los aspirantes a distintos concursos docentes en la UTN. La información sobre cada concurso se representa así:

```
concursoPdp =
    [("pepe",[7,4,6,3]), ("cacho",[4,6,9,2]), ("pichu",[3,5,8,7]),
     ("rodro",[9,5,8,6])]
concursoTrucho =
    [("capo",[7,4,0,3]), ("pichu",[15,4,9,2]), ("jose",[3,5,3,5])]
```

Los cuatro números para cada aspirante son, en orden: prueba de oposición, antigüedad, antecedentes académicos, y antecedentes profesionales.

1.

Definir la función calificaciones, que dados una persona y un concurso, devuelve la lista de calificaciones de la persona en ese concurso. P.ej.

```
calificaciones "cacho" concursoPdp
```

devuelve `[4,6,9,2]`. En este ítem no se puede usar recursividad.

2.

a. Definir la función bienCargada, que dada una lista de números, devuelve True si todos están entre 1 y 10 (calificaciones válidas), y False en caso contrario. P.ej.

`bienCargada [7,4,25,3]` devuelve `False`, mientras que
`bienCargada [4,6,3,2]` devuelve `True`

Usar recursividad. Ayuditas:

- sobre la lista vacía se espera que devuelva True,
- tener en cuenta las funciones elem y &&

b. Definir la función malCargados, que dado un concurso devuelve los nombres de aquellos cuyas calificaciones no están bien cargadas. P.ej.

```
malCargados concursoTrucho
```

devuelve `["capo","pichu"]`. Usar bienCargada.

3.

a. Definir la función maximasCalificaciones, que dado un concurso devuelve la lista de pares (aspirante, calificación más alta obtenida por el aspirante). P.ej.

```
maximasCalificaciones concursoPdp
```

devuelve `[("pepe",7),("cacho",9),("pichu",8),("rodro",9)]`.
Usar listas por comprensión.

b. Definir la función puntajesPruebaOposicion, que dado un concurso devuelve la lista de pares (aspirante, calificación del aspirante en la prueba de oposición). P.ej.

```
puntajesPruebaOposicion concursoPdp
```

devuelve `[("pepe",7),("cacho",4),("pichu",3),("rodro",9)]`

Usar listas por comprensión. Ayuda: la prueba de oposición es el primer elemento de la lista de calificaciones.

4.

a. Definir la funcion aceptados, que dados una función, un valor de corte y un concurso, devuelve los nombres de los aspirantes que se aceptarían como posibles docentes tomando como criterio la función evaluada sobre las calificaciones de cada aspirante.
P.ej. en

```
aceptados minimoList 4 concursoPdp
```

la respuesta debe ser `["rodro"]`, porque los mínimos de las calificaciones de pepe, cacho, pichu y rodro son 3, 2, 3 y 5 respectivamente, y el único que pasa el corte de 4 es rodro.
Si la función elegida es el puntaje obtenido en la prueba de oposición, y el valor de corte es 7, los aceptados serán pepe y rodro.
OJO - no se puede usar ni recursividad ni listas por comprensión.
Indicar el tipo de la función definida.

b. Indicar cómo usaría la función aceptados para estos criterios:
- valor en la prueba de oposición, corte 7
- suma de las calificaciones, corte 15
- antigüedad, corte 5 (definir una función auxiliar antiguedad).

5.

Definir la función aceptadosPorAlguno, que recibe dos funciones, un valor de corte y un concurso, y devuelve los nombres de los aspirantes que son aceptados (según la definición del punto a) según al menos una de las dos funciones. P.ej.

```
aceptadosPorAlguno head antiguedad 6 concursoPdp
```

debe devolver `["pepe","cacho","rodro"]`.
Debe usarse la función aceptados. Ayudas: expresión lambda, función min que devuelve el mínimo entre dos números (p.ej. `min 5 3` devuelve `3`).

6.

a. Definir la función valorSegun, que dados una función, un aspirante y un concurso devuelve el valor para la función evaluada sobre las calificaciones del aspirante en ese concurso. P.ej.

```
valorSegun sum "cacho" concursoPdp
```

devuelve `21`.

b. Definir la funcion esMejor, que dados dos aspirantes, un concurso y una función, indica si el primero tiene un valor mayor para esa función que el segundo o no. P.ej.

```
esMejor "rodro" "cacho" concursoPdp sum
```

devuelve `True` porque el valor de sum para rodro es 28, que es mayor a 21 (valor para cacho).

c. Definir la funcion mejorEnTodo, que dados una lista de funciones, dos aspirantes y un concurso, indica si el primero es mejor en todos los criterios indicados que el segundo. Ejemplo de uso

```
mejorEnTodo lista_de_funciones "cacho" "rodro" concursoPdp
```

Indicar cómo la usaría (qué pondría en lista_de_funciones) para comparar a cacho y rodro según suma de las calificaciones, calificación de la prueba de oposición, y antigüedad.

7.

a. Definir la funcion maximoFn, que dados una función y una lista, devuelve el elemento de la lista que hace máxima la función. P.ej.

```
maximoFn abs [-5..3]
```

devuelve `-5`.

b. Definir la funcion ganador, que dados una función y un concurso, indica el nombre del aspirante que obtuvo máximo puntaje según la función indicada. P.ej.

```
ganador sum concursoPdp
```

devuelve "`rodro`", que es el aspirante a ese concurso que obtuvo mejor puntaje si se tiene en cuenta la suma de las calfiicaciones obtenidas.