Práctica 8: Repaso para 2da Prueba

1. Considere la definición de un nuevo tipo de dato que llamaremos `('a, 'b) elemento` :

```
- datatype ('a,'b) elemento =
            P of 'a * 'b |
            S of 'a;
datatype ('a,'b) elemento = P of 'a * 'b | S of 'a
```

Este tipo de dato nos permite representar elementos simples, al usar el constructor `S`, y pares ordenados, al usar el constructor `P`. Los tipos `'a` y `'b` pueden ser cualquier tipo de dato que se desee.

En particular, vamos a elegir que `'a` sea `string` y `'b` sea `int` y vamos a usar de tipo de dato para analizar un texto y contar la aparición de ciertas palabras de interés. Aquellas palabras que se encuentren en el texto las representaremos mediante pares, donde la primer componente será la palabra y la segunda componente la cantidad de veces que aparece en el texto y aquellas que no se encuentren en el texto las representaremos mediante elementos simples.

Ejemplo: Supongamos que al analizar un texto nos interesa conocer la cantidad de apariciones de las siguientes palabras: *en*, *función*, *como*, *después* y *antes*, y al analizar el texto obtenemos la siguiente lista:

`[P("en", 10), S("función"), P("como",2), S("después"), P("antes", 1)]`

La lista obtenida signfica que: la palabra *en* aparece 10 veces, *función* no aparece, *como* aparece 2 veces, *después* no aparece y *antes* sólo 1 vez.

Necesitamos definir funciones que nos ayuden en el análisis de este tipo de listas de elementos. Se pide entonces que defina las siguientes funciones:

   a. Defina una función `aparece` que tome una lista de `(string, int) elemento` obtenida luego del análisis de un texto y devuelva aquellas palabras que han sido encontradas en el texto analizado, o sea, una lista de aquellas palabras que forman parte de un par ordenado. *Ejemplo:* `aparece([P("en", 10), S("función"), P("como",2), S("después"), P("antes", 1)]) = ["en", "como","antes"]`

   b. Defina una función `cant-aparece` que tome una lista de `(string, int) elemento` obtenida luego del análisis de un texto y cuente cuántas ocurrencias hay en el texto analizado de las palabras buscadas. *Ejemplo:* `cant-aparece([P("en", 10), S("función"), P("como",2), S("después"), P("antes", 1)]) = 13`

   c. Defina una función `no-aparece` que tome una lista de `(string, int) elemento` obtenida luego del análisis de un texto y devuelva aquellas palabras que NO han sido encontradas en el texto analizado, o sea, una lista de aquellas palabras que son elementos simples. *Ejemplo:* `no-aparece([P("en", 10), S("función"), P("como",2), S("después"), P("antes", 1)]) = ["función", "después"]`

   d. Defina una función `cant-no-aparece` que tome una lista de `(string, int) elemento` obtenida luego del análisis de un texto y cuente cuántas de las palabras buscadas NO han sido encontradas en el texto analizado. *Ejemplo:* `cant-no-aparece([P("en", 10), S("función"), P("como",2), S("después"), P("antes", 1)]) = 2`

2. Considere la siguiente definición de árbol binario ya vista en clase:

```
- datatype 'etiqueta arbolbin =
            Vacio |
            Nodo of 'etiqueta * 'etiqueta arbolbin * 'etiqueta arbolbin;
datatype 'a arbolbin = Nodo of 'a * 'a arbolbin * 'a arbolbin | Vacio
```

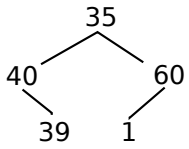
Suponga que existe un árbol binario de búsqueda implementado usando el tipo de dato `(string, int) arbolbin` que representa la lista de estudiantes de *Programación II* con sus respectivas notas. Implemente las siguientes funciones que ayuden a la profesora en su trabajo:

a. Defina una función `ingresar-nota` que tome un árbol binario de búsqueda `(string, int) arbolbin` que contenga datos de alumnos, un nombre de un alumno y una nota para dicho alumno y si ese nombre aparece ya en el árbol deberá cambiarle la nota, si no aparece deberá agregar el alumno con su respetiva nota.

b. Defina una función `buscar-nota` que tome un árbol binario de búsqueda `(string, int) arbolbin` que contenga datos de alumnos, un nombre de un alumno y devuelva la nota de dicho alumno si ese nombre aparece en el árbol, si no aparece deberá devolver `~1`.

c. Defina una función `definir-aprobacion` que tome un árbol binario de búsqueda `(string, int) arbolbin` que contenga datos de alumnos y devuelva otro árbol binario de búsqueda de tipo `(string, string) arbolbin` donde cada nodo del árbol contenga el nombre del alumno acompañado de `"Aprobado"` si la nota es mayor o igual a 6 o `"No Aprobado"` si la nota del alumno es inferior a 6.

3. Dado el siguiente árbol:



represéntelo usando:

a. el siguiente tipo de dato:

```
- datatype int arbolbin =
             Vacio |
             Nodo of int * int arbolbin * int arbolbin;
```

b. el siguiente tipo de dato:

```
- datatype int arbol =
             Nodo of int * int arbol list;
```

4. Considere la siguiente definición de árbol binario ya vista en clase:

```
- datatype 'etiqueta arbolbin =
            Vacio |
            Nodo of 'etiqueta * 'etiqueta arbolbin * 'etiqueta arbolbin;
```

Suponga que existe una agenda telefónica implementada como un árbol binario de búsqueda de tipo `(string, string, int) arbolbin`. Cada nodo del árbol es una terna que contiene el nombre de una persona (escrito todo en minúscula), la dirección y el teléfono. El árbol binario de búsqueda está ordenado de acuerdo al orden alfabético del campo nombre. Implemente las siguientes funcionalidades que permitan utilizar la agenda:

a. Defina una función `ingresar-contacto` que tome una agenda (implementada como un árbol binario de búsqueda) y los datos de una persona (nombre, dirección y teléfono) y si dicha persona no se encuentra ya en la agenda se agrega a la misma (recuerde mantener la propiedad de árbol binario de búsqueda). Si la persona ya se encuentra en la agenda no debe modificar la agenda.

b. Defina una función `modificar-contacto` que tome una agenda (implementada como un árbol binario de búsqueda) y los datos de una persona (nombre, dirección y teléfono) y si dicha persona se encuentra ya en la agenda modifica los datos asociados. Si la persona no se encuentra en la agenda no debe modificar la agenda.

c. Defina una función `buscar-direccion` que tome una agenda (implementada como un árbol binario de búsqueda) y el nombre de una persona y devuelva su dirección en caso de que dicha persona esté en la agenda. Si la persona no se encuentra en la agenda debe devolver el mensaje "No se encontró el contacto buscado".

d. Defina una función `buscar-telefono` que tome una agenda (implementada como un árbol binario de búsqueda) y el nombre de una persona y devuelva su teléfono en caso de que dicha persona esté en la agenda. Si la persona no se encuentra en la agenda debe devolver `~1`.

Para resolver algunos de los siguientes ejercicios deberá utilizar las funciones de alto orden que vienen definidas ya en ml: `map`, `foldl`, `foldr` y `filter`. No debe redefinirlas. Para resolver algunos de los ejercicios deberá utilizar, además, una combinación de ellas.

5. Dado el siguiente tipo de dato:

```
- datatype 'etiqueta arbolbin =
            Vacio |
            Nodo of 'etiqueta arbolbin * 'etiqueta * 'etiqueta arbolbin;
```

   a. Defina la función `preOrden` que realice el recorrido *preorden* de un `arbolbin`

   b. Defina la función `inOrden` que realice el recorrido *inorden* de un `arbolbin`

   c. Defina la función `postOrden` que realice el recorrido *postorden* de un `arbolbin`

6. Suponga que representamos puntos en el plano con pares ordenados de reales. O sea, definimos el tipo de dato par:

```
- type par = real * real;
type par = real * real
```

   Un ejemplo de un punto en el plano sería el siguiente:

```
- val p=(2.0,3.0):par;
val p = (2.0,3.0) : par
```

   Sabemos que si tenemos un punto (x,y) en el plano, su distancia al origen de coordenadas se calcula así:

$$dist\_origen\_coordenadas(x, y) = \sqrt{x^2 + y^2}$$

   a. Defina la función `listdist` que tome una lista de puntos del plano y devuelva la lista de sus distancias al origen de coordenadas.

   b. Defina la función `sumdist` que tome una lista de puntos del plano y devuelva la suma de sus distancias al origen de coordenadas.

   c. Defina la función `puntosalejados` que tome una lista de puntos del plano y devuelva otra lista que contenga sólo aquellos puntos que se encuentran a una distancia estrictamente mayor a `5.0` del origen de coordenadas.

7. Considere nuevamente la definición de árbol binario del ejercicio **??**.

   Suponga que existe una lista de estudiantes con sus notas de un parcial implementada como un árbol binario de búsqueda de tipo `(string * int) arbolbin`. Cada nodo del árbol es una tupla que contiene el nombre del alumno y su nota. El árbol binario de búsqueda está ordenado de acuerdo al orden alfabético del campo nombre (suponga todos los nombres escritos en minúscula y sin acentos). Implemente las siguientes funciones:

   a. Defina una función `listar_en_orden_creciente` que tome la lista de estudiantes implementada como un árbol binario de búsqueda `(string * int) arbolbin` y devuelva un `(string * int) list` de todos los alumnos con sus notas. Use alguna de las funciones de recorrido definidas en el ejercicio **??**.

   b. Defina una función `buscar_sobresalientes` que tome la lista de estudiantes implementada como un árbol binario de búsqueda `(string * int) arbolbin` y devuelva un `(string, int) list` de todos los alumnos que hayan obtenido un 10 como calificación. Debe usar la función `listar_en_orden_creciente` y `filter` para su definición.