Práctica 6: Repaso examen sobre datatypes

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 $\sim$`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.