Instituto Politécnico Superior "Gral. San Martín"
Departamento de Informática
5^{to} Año - Programación II

Nombre y Apellido: Mail:

Prueba
Tema 1

Para resolver este examen debe completar la hoja adjunta que se le entrega (que contiene porciones de código). No puede cambiar el código que se le provee ni las signaturas de las funciones, sólo debe completar las partes que faltan. Puede definir funciones auxiliares si lo considera necesario.

1. Trabajaremos con la siguiente estructura:

```
type tiempo = {hs: int, min: int, segs: int};
```

a. Defina la función `tiemposCorrectos` que tome una lista de estructuras tiempo (`tiempo list`) y devuelva `true` en caso de que todos los elementos de la lista representen horas de un día, `false` en caso contrario. Para que la hora se considere correcta deberá tener un valor entre 0 y 23 en el campo `hs` y valores entre 0 y 59 en los campos `min` y `segs`. *Ejemplos:*
`val hora1 = {hs=23, min=55, segs=45}:tiempo;`
`val hora2 = {hs=12, min=0, segs=0}:tiempo;`
`val hora3 = {hs=23, min=75, segs=25}:tiempo;`
`tiemposCorrectos([hora1, hora2]) = true` (ya que ambas horas respetan los límites).
`tiemposCorrectos([hora1, hora3]) = false` (ya que `hora3` no respeta el límite establecido para el campo `min`).
Función auxiliar: debe definir una función auxiliar `tiempoCorrecto` que tome una estructura `tiempo` y devuelva `true` o `false` dependiendo si esa estructura representa una hora correcta o no.
Ejemplos: `tiempoCorrecto(hora1) = true`, `tiempoCorrecto(hora2) = true`, `tiempoCorrecto(hora3) = false`

b. Defina la función `listaSegsMedianoche` que tome una lista de estructuras tiempo y devuelva por cada estructura que representa una hora correcta la cantidad de segundos que transcurrieron desde la medianoche y por cada estructura que no representa una hora correcta el valor $\sim$1.
Ejemplo: `listaSegsMedianoche[hora1, hora2, hora3] = [86145, 43200, ~1]`.
Función Auxiliar: debe definir una función auxiliar `segsMedianoche` que tome una estructura tiempo (`tiempo list`) que representa un tiempo y devuelva la cantidad de segundos que pasaron desde la medianoche. *Ejemplos:* `segsMedianoche(hora1) = 86145` , `segsMedianoche(hora2) = 43200`. *No hace falta que tenga en cuenta el caso en el cual el par ordenado no represente una hora posible.*

c. Defina la función `listaSupera75000` que tome una lista de estructuras `tiempo` y devuelva otra lista que contenga sólo aquellas estructuras que representan una hora correcta y que distan en más de 75000 segundos de la medianoche.
Ejemplo: `listaSupera75000([hora1, hora2, hora3])=[hora1]`

2. Considere la siguiente definición de árbol binario ya vista en clase y un nuevo tipo de dato `auto`:

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

type auto = {Marca: string, Modelo: string, Colores: string list, PrecioBase: real};
```

Suponga que una concesionaria guarda el registro de todos sus autos como un árbol binario de búsqueda de tipo `auto arbolbin` (existe un único registro por modelo de auto). Cada nodo del árbol es una estructura que contiene los datos de un modelo (el modelo del auto escrito todo en minúscula), su marca, su precio base y una lista de colores posibles. El árbol binario de búsqueda está ordenado de acuerdo al orden alfabético del campo `Modelo`. Implemente las siguientes funcionalidades que permitan utilizar la agenda:

a. Defina una función `ingresar_auto` que tome el registro de todos los autos de esta concesionaria (implementada como un árbol binario de búsqueda) y los datos de un auto y si dicho auto no se encuentra ya en el registro se agrega al mismo (recuerde mantener la propiedad de árbol binario de búsqueda). Si el auto ya estaba en el registro deben actualizarse sus datos.

Ayuda: Para esto defina antes una función `crear_auto` que dados los datos de un auto, cree la estructura auto con dichos datos.

b. Defina una función `cambiar_precio` que tome el registro de autos (implementado como un árbol binario de búsqueda), un modelo y un precio y si dicho auto se encuentra en el registro, modifique su precio base. Si el modelo de auto no se encuentraba en el registro no debe modificar nada en el mismo.

c. Defina una función `buscar_colores` que tome el registro de autos (implementado como un árbol binario de búsqueda) y un modelo de auto y devuelva devuelva la lista de colores disponible. Si dicho auto no se encuentra en el registro debe devolver la lista vacía.

d. Defina una función `buscar_autos` que tome el registro de autos (implementado como un árbol binario de búsqueda) y un color y devuelva la lista de autos (`auto list`) que la concesionaria dispone en dicho color (puede que no haya ningún auto disponible).

e. Defina una función `buscar_modelos` que tome el registro de autos (implementado como un árbol binario de búsqueda) y un color y devuelva la lista de modelos (`string list`) que la concesionaria dispone en dicho color (puede que no haya ningún modelo de auto disponible).

f. Defina una función `cantidad_disponibles` que tome el registro de autos (implementado como un árbol binario de búsqueda) y un color y devuelva la cantidad de modelos de autos que se encuentran disponibles en dicho color (puede ser cero).