Segundo Parcial – Tema 1

- **NO OLVIDAR PONER EL NOMBRE Y APELLIDO EN CADA HOJA**
- **NUMERAR LAS HOJAS CON LA FORMA <número de hoja>/<cantidad total de hojas>**
- Leer el examen completo antes de empezar a resolver los ejercicios, ya que ayuda a comprender mejor el dominio. Se puede consultar cualquier material (en papel) que haya sido escrito antes de comenzar el examen y usar sin definir todas las funciones y procedimientos vistos durante la cursada.
- Pensar bien la estrategia a seguir, consultar lo que no se entienda y recordar que el parcial es una instancia más de aprendizaje donde algún docente va a dar una devolución personalizada de todo lo que se escriba, es necesario para aprovechar esta instancia como algo más que solamente un momento para sacar una nota.

Refugio de Animales

Una serie de refugios de animales quiere incorporar algunas funcionalidades a su sistema para agilizar sus tareas. Actualmente su sistema cuenta con los siguientes tipos:

```
type Refugio is record { /*
  PROPÓSITO: Modelar un refugio de animales.
  INV. REP.:
    * La lista **animales** no tiene animales con el mismo nombre.
    * La lista **cuidadores** no tiene cuidadores con el mismo nombre.
    * Todo animal de la lista **animales** tiene asignado uno o más
        cuidadores de la lista **cuidadores**.
    * Cada nombre de animal asignado a un cuidador de la lista
        **cuidadores** corresponde al nombre de un animal de la lista
        **animales**.
  */
  field animales   // [Animal]
  field cuidadores // [Cuidador]
}
```

```
type Especie is
    variant { /*
  PROPÓSITO: Modelar
    las especies de
    animales del
    refugio.
  */
  case Yaguareté {}
  case Vicuña    {}
  case Yarará    {}
  case Hornero   {}
  case Condor    {}
  case Carpincho {}
}
```

```
type Cuidador is record { /*
  PROPÓSITO: Modelar un cuidador de animales.
  INV.REP.:
    * **nombre** no es un string vacío.
    * La lista **nombresACargo** no tiene
        repetidos.
  */
  field nombre           // String
  field nombresACargo    // [String]
}
```

```
type Animal is record { /*
  PROPÓSITO: Modelar un animal.
  INV. REP.:
    * **nombre** no es un string vacío.
    * **peso** es mayor a cero.
*/
  field nombre  // String
  field especie // Especie
  field peso    // Número
}
```

Usando este modelo como base, considere las siguientes situaciones:

Punto 1)

Cuando un animal es liberado nuevamente a la naturaleza, se debe quitar todo registro del mismo del refugio donde estaba. Esto implica no solo quitarlo de la lista de animales del refugio, sino también de las listas de animales a cargo de todos aquellos cuidadores que pudieran tener dicho animal a su cuidado. Se pide, por ello, que implemente la función **refugio_SinAnimalLlamado_** que dados un refugio y un string que representa el nombre de un animal que existe en ese refugio, describa el refugio que resulta de remover el animal con dicho nombre del refugio dado.

Pista: Recuerde qué funciones de biblioteca tiene disponibles para usar al procesar listas.

Punto 2)

Tanto los cambios de refugio como los cambios de cuidador son estresantes para los animales. Por eso, a la hora de decidir dónde ubicar a un animal nuevo se busca aquel refugio que tenga el mayor número de cuidadores con experiencia en su especie. Se pide entonces que implemente la función **mejorRefugioEntre_ParaAnimal_** que dados una lista de refugios y un animal, describa a aquel refugio de entre los dados que tiene el mayor número de cuidadores con experiencia en su especie. Decimos que un cuidador tiene experiencia en una especie si actualmente cuida al menos un animal de dicha especie.

Se sugiere que su estrategia incluya la implementación de la subtarea **animalesDeEntre_NombradosEn_** que dada una lista de animales y una lista de strings que representan nombres de animales, describe una nueva lista conformada por aquellos animales de la primera lista cuyos nombres aparecen en la segunda lista. **Aclaración**: esta subtarea es sólo una sugerencia y, en caso de optar por usarla DEBE implementarla ya que no se propone como primitiva.

Pista: Recuerde qué funciones de biblioteca tiene disponibles para usar al procesar listas.

Punto 3)