UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2012

01 de Octubre de 2012

TPF - Hoteles Funcionales v1.3

1. Introducción

En el TPE, hemos especificado el comportamiento de una serie de problemas relacionados con el mundo de la hotelería. Nuestro próximo paso será obtener un programa funcional que respete la especificación que adjunta la cátedra para esta nueva etapa del proyecto. El objetivo de este trabajo es, entonces, programar en Haskell los tipos y operaciones del TPE. A continuación diremos cómo decidimos implementar cada tipo y qué funciones exporta (interfaz). La interfaz de los tipos no podrá ser modificada, salvo por expresa (expresa==escrito) indicación de la cátedra. En cada tipo pueden implementar su propio `show`, de tal manera que se sientan cómodos con su visualización. **NO** pueden agregar el `Eq` si el mismo no se encuentra en el presente enunciado. Se pueden agregar funciones auxiliares dentro de cada módulo, pero **NO** se pueden exportar.

Para la resolución del trabajo, se pueden usar únicamente las funciones y operadores `last`, `init`, `head`, `tail`, `!!`, `reverse`, `++`, `elem`, `length`, `fst`, `snd` y los operadores de comparación entre elementos de un mismo tipo. Notar que la especificación que se adjunta **NO** es una solución al tp de especificación.

Los tipos **NO** pueden ser transformados en listas para la implementación de los problemas.

Su trabajo consistirá en implementar todos los problemas especificados por la cátedra (ver archivo aparte). A continuación se detalla la definición de los módulos que deben implementar.

2. Módulo Tipos

```
module Tipos where

type Fecha = Int
type DNI = Int
type Dinero = Int
type Cadena = String
type Nombre = String
type Provincia = String
type CheckIn = (DNI, Fecha)
type CheckOut = (DNI, Fecha)

data Sexo = Femenino | Masculino deriving (Show, Eq)

data TipoHabitacion = Simple | Doble | Triple | Cuadruple deriving (Show, Eq)
data Accesorio = Jacuzzi | LCD | PS3 | DVD | Pelotero | Inflable deriving (Show, Eq, Ord)
```

3. Módulo Habitacion

```
module Habitacion (Habitacion, nuevaH, numeroH, tipoH, accesoriosH, pretensionesDePopStarH)

where

import Tipos

data Habitacion = A Accesorio Habitacion | H Int TipoHabitacion deriving (Show)
```

donde el constructor `A` recibe los parámetros en el siguiente orden:

1. Accesorio a agregar en la habitacion
2. Habitacion en la que se agrega el accesorio

Y donde el constructor `H` recibe los parámetros en el siguiente orden:

1. Número de la habitación
2. Tipo de habitacion

4. Módulo Reserva

```
module Reserva (Reserva, nuevaR, documentoR, fechaDesdeR, fechaHastaR, tipoR, confirmadaR, confirmarR)

where

import Tipos

data Reserva = R DNI Fecha Fecha TipoHabitacion Bool deriving (Show, Eq)
```

donde el constructor `R` recibe los parámetros en el siguiente orden:

- DNI de quien realiza la reserva
- fecha desde
- fecha hasta
- tipo de habitación
- true si está confirmada, false en caso contrario

5. Módulo Hotel

```
module Hotel (Hotel, sobreReservadoH, huespedesConPalabraH, nuevoH,
                nombreH, cadenaH, ingresosH, salidasH, huespedesH, reservasH, habitacionesH,
                tarifaXDiaH, precioAccesorioH, registrarHuespedH, desregistrarHuespedH, hacerReservaH,
                calcularCuentaH, reservasSolapadasH, venderH)

where

import Tipos
import Habitacion
import Reserva
import List

data Hotel =    HT Nombre Cadena [Habitacion] [(TipoHabitacion, Dinero)] [(Accesorio, Dinero)]
                    | CI DNI Fecha Habitacion Hotel
                    | CO DNI Fecha Hotel
                    | R Reserva Hotel
                    deriving (Show)
```

donde:

- el constructor `HT` permite crear una instancia de Hotel y los parámetros representan (en el mismo orden):
  - el nombre del hotel
  - la cadena a la cual el hotel pertenece
  - las habitaciones del hotel
  - la lista de precios de las habitaciones según su tipo
  - la lista de precios de los accesorios de las habitaciones
- el constructor `CI` permite registrar un ingreso en el hotel y los parámetros representan (en el mismo orden):
  - DNI de quien realizar el ingreso
  - Fecha en la que se realiza el ingreso
  - Habitación en la que se registra el huesped

- Hotel en cuestión

- el constructor `CO` permite registrar una salida del hotel y los parámetros representan (en el mismo orden):
  - DNI de quien realizar la salida
  - Fecha en la que se realiza la salida
  - Hotel en cuestión
- el constructor `R` registra una nueva reserva y los parámetros representan (en el mismo orden):
  - Reserva a registrar
  - Hotel en cuestión

6. Módulo MinisterioDeTurismo

```
module Ministerio (Ministerio, agregarHotelM, secretariasM,
            registroM, nuevoM, cadenasDeHotelesM, cadenasAmarretasM, fusionAutorizadaM)

where

import Tipos
import Reserva
import Habitacion
import Hotel

data Ministerio =  REG Provincia Hotel Ministerio |  MT [Provincia] deriving (Show)
```

donde:

- el constructor `MT` permite crear una instancia de MinisterioDeTurismo y su parámetro representa la lista de provincias registradas.
- el constructor `REG` permite representar los hoteles registrados en cada una de las provincias y sus paramámtros representan (en el mismo orden):
  - Provincia donde se registrará el hotel
  - Hotel a registrar
  - Ministerio en cuestión.