Trabajo Práctico # 1

Estructuras de Datos, Universidad Nacional de Quilmes

12 de marzo de 2013

1. Listas

Ejercicio 1

Defina las siguientes funciones sobre listas, usando el esquema de recorrido recursivo:

1. `mapSucesor`: dada una lista de enteros, devuelve la lista de los sucesores de cada entero.
2. `filterPositivos`: dada una lista de enteros, devuelve una lista con los elementos que son positivos.
3. `filterPrimos`: dada una lista de enteros, devuelve la lista de los que son primos. *Nota: un número primo es un número natural mayor que 1 que tiene únicamente dos divisores distintos: él mismo y el 1.*
4. `zipMaximos1`: dadas dos listas de enteros, devuelve una lista donde el elemento n es el máximo entre el elemento n de la lista 1 y de la lista 2 teniendo en cuenta que las listas tienen siempre la misma longitud.
5. `zipMaximos2`: dadas dos listas de enteros, devuelve una lista donde el elemento n es el máximo entre el elemento n de la lista 1 y de la lista 2 teniendo en cuenta que las listas no necesariamente tienen la misma longitud.
6. `zipSort`: dadas dos listas de enteros de igual longitud, retorna una lista de pares (min, max), donde min y max son el mínimo y el máximo entre los elementos de ambas listas en la misma posición.
7. `mapLongitudes`: dada una lista de listas, devuelve la lista de sus longitudes.
8. `longitudMayorA`: dada una lista de listas y un número n, devuelve la lista de aquellas listas que tienen más de n elementos.
9. `mapCuadrados`: dada una lista de enteros, devuelve la lista de los cuadrados de los elementos positivos, en el mismo orden.
10. `sumaPar`: dada una lista de pares, devuelve una nueva lista en la que cada elemento es la suma de los elementos de cada par.
11. `takePersonas`: dada una lista de Personas [nombre, apellido y fecha de nacimiento] ordenada ascendentemente por fecha de nacimiento; y una fecha, devuelve el segmento más largo de la lista con las personas que nacieron antes dicha fecha. Tanto el nombre como el apellido se pueden representar con el tipo de dato `Char`. La fecha se sugiere que sea de la forma (año,mes,día).
12. `dropPrecio`: dada una lista de Pizzas [lista de ingredientes y precio] en orden ascendente por precio, devuelve el segmento más largo de la lista que comienza con la pizza que tiene el menor precio superior a $30.

13. `takeNombresPersonas`: dada una lista de Personas y una fecha devuelve los nombres de las personas incluidas en segmento más largo de la lista con las personas que nacieron antes dicha fecha.

14. `sumParesColor`: dadas dos listas de ParColor [color y cantidad], devuelve una lista de ParColor donde la cantidad de cada elemento es la suma del elemento de ese color en cada lista. *Nota: las dos listas no necesariamente tienen los mismos colores. Asumir cantidad 0 para colores ausentes.*

15. `reversa`: dada una lista de enteros, devuelve la lista con los mismos elementos de atrás para adelante.

16. `aplanar`: dada una lista de listas, devuelve una única lista con todos sus elementos.

17. Clasifique las funciones definidas en diversos grupos según sus similitudes.

Ejercicio 2

Defina las siguientes funciones sobre listas, usando el esquema de recorrido recursivo:

1. `append`: dadas dos listas devuelve la lista con todos los elementos de la primer lista y todos los elementos de la segunda a continuación.

2. `snocAlt`: dada una lista de enteros xs y un entero i, devuelve la lista resultante de agregar i al final de xs. *Nota: utilizar* ***Cons****.*

3. `digs2Num10`: dada una lista de dígitos, devuelve el número que representa en base 10.
   Para calcular el número y representado por una secuencia de dígitos $[x_1, \ldots, x_n]$ en base b, utilizar la siguiente fórmula:
   $$y = \sum_{i=1}^{n} b^{n-i} x_i$$
   Nota: Aquí utilizamos la sumatoria, la cual es un operador matemático que permite representar sumas de muchos sumandos, n o incluso infinitos sumandos, se expresa con la letra griega sigma .

4. Defina `digs2Num8`, `digs2Num2` y `digs2Num9`, que transforman una secuencia de dígitos ds en el número que representa ds en las bases 8, 2 y 9 respectivamente. ¿Cuál es la precondición que deben satisfacer los dígitos de ds en cada caso?

Ejercicio 3

Defina las siguientes operaciones parciales, usando el esquema de recorrido recursivo. Defina también las precondiciones necesarias para poder utilizarlas.

1. `init`: dada una lista, devuelve una copia de esta sin su último elemento.

2. `last`: dada una lista, devuelve su último elemento.

3. `lookUp`: dada una lista de strings y un string s devuelve la posición de la lista recibida en la que se encuentra s.

4. `takeN`: dada una lista x y un número n, devuelve una lista con los primeros n elementos de lista recibida. Si la lista recibida tuviera menos de n elementos, devuelve la lista completa.

5. `dropN`: dada una lista y un número n, devuelve una lista sin los primeros n elementos de lista recibida. Si la lista recibida tuviera menos de n elementos devuelve la lista vacía.

6. `valorParaClave`: dada una lista de pares (clave, valor) y una clave devuelve el valor asociado a la clave.

7. `maximum`: dada una lista de enteros devuelve el máximo.

Ejercicio 4

Defina las versiones totales de las operaciones parciales definidas en el punto anterior, usando el esquema de recorrido recursivo y empleando el tipo de dato `Maybe` donde corresponda.

1. `mLookUp`: dada una lista de strings y un string s devuelve la posición de la lista recibida en la que se encuentra s o `Nothing` si s no se encuentra en la lista.
   ¿Que diferencia existe entre usar mLookUp y lookUp? ¿Que situaciones me permite manejar cada una?

2. `mInit`: dada una lista, devuelve una copia de esta sin su último elemento o `Nothing` si la lista está vacía.

3. `mLast`: dada una lista, devuelve su último elemento o `Nothing` si la lista está vacía.

4. `takeExactlyN`: dada una lista y un número n, devuelve una lista con los primeros n elementos de lista recibida. Si la lista recibida tuviera menos de n, elementos devuelve `Nothing`.

5. `dropExactlyN`: dada una lista y un número n, devuelve una lista sin los primeros n elementos de lista recibida. Si la lista recibida tuviera menos de n elementos, devuelve `Nothing`.

6. `mValorParaClave`: dada una lista de pares (clave, valor) y una clave devuelve el valor asociado a la clave si existiera. En caso contrario, devuelve `Nothing`.

7. `mMaximo`: dada una lista de enteros devuelve el máximo. Si la lista está vacía, devuelve `Nothing`.