CC4101- Lenguajes de Programación
Auxiliar 2

Richard Ibarra Ramírez

4 de abril de 2010

1. Para reforzar los conceptos de funciones de primera clase y recursión, implementaremos QuickSort en Scheme.

   a) QuickSort sobre números

   - Defina la función `filter-<-pivot`.
     `;; filter-<-pivot :: number, list<number> -> list<number>`
   - Análogamente implemente la funcion `filter->=-pivot`.
   - Defina QuickSort usando las funciones `filter-*-pivot`.
   - ¿Es ésta una versión *inteligente*? ¿Qué pasa si ahora quiero definir QuickSort sobre otro tipos de datos (strings, vectores, etc)?

   b) QuickSort genérico

   - Defina la función filter-pivot.
     `;; filter-pivot :: X (X X -> boolean) list<X> -> list<X>`
   - Defina quicksort usando `filter-pivot`.
     Hint. *Para operar sobre los elementos que fueron 'filtrados', ocupe una función anónima que niegue el resultado del comparador.*

   c) QuickSort funcional

   Si bien la versión cumple su objetivo y puede ser reutilizado en varías situaciones, QuickSort puede ser definido genéricamente como un filtrado sucesivo de elementos.

   - Defina QuickSort usando `filter` y recibiendo como parámetros un comparador y una lista de elementos.

2. Considere la siguiente definición BNF

```
<BinTree> := ()
           | (<Symbol> . <BinTree> . <BinTree>)
```

   a) Defina la estructura `BinTree` usando `define-type`.

   b) Defina la función `sum-nodes` que dado un `BinTree` cuyos nodos son sólo números, retorna la suma de todos ellos.

c) Defina la función `make-bintree` que dado una lista de números retorna el árbol resultante de insertar cada uno en orden ascendente.

d) Defina la función `map-bintree` que retorna un nuevo `BinTree` resultante de aplicar una función en cada uno de los nodos.