v.6.12

Programación 1 - Segundo Parcial - Tema 1

(19-06-2018)

1 Listas, primera parte

Ejercicio 1. Diseñe una función `cuantos-mayores-a` que dada una lista de enteros `l` y un entero `x` devuelva la cantidad de elementos de `l` que son mayores a `x`.

Por ejemplo:

```
(check-expect (cuantos-mayores-a 5 (list 1 6 8 0 8)) 3)
(check-expect (cuantos-mayores-a 3 (list 0 1 2 3)) 0)
```

Incluya en su diseño al menos dos ejemplos adicionales. No utilice las funciones `map`, `foldr` ni `filter`.

2 Listas, segunda parte

Ejercicio 2. Decimos que dos números naturales (a, b) son *catetos pitagóricos* si en el triángulo rectángulo con catetos de longitudes a y b la hipotenusa es un número entero. Es decir, si $\sqrt{a^2 + b^2}$ es entero. Por ejemplo, los números (3,4) son *catetos pitagóricos*; mientras que (1,1) no.

Para un par de números (a,b) que son catetos pitagóricos, al valor entero de la expresión $\sqrt{a^2 + b^2}$ le llamamos el *pitagórico mayor* del par. Por ejemplo, el *pitagórico mayor* del par (3,4) es 5.

Diseñe una función `f` que dada una lista `l` de estructuras `posn`, devuelva la suma de los *pitagóricos mayores* de los pares que son *catetos pitagóricos*.

Por ejemplo:

```
(check-expect (f (list (make-posn 4 3) (make-posn 1 2) (make-posn 7 24))) 30)
(check-expect (f (list (make-posn 4 4) (make-posn 1 2))) 0)
```

Incluya en su diseño otros ejemplos adecuados. Utilice en sus definiciones las funciones `map`, `foldr` y/o `filter` según sea necesario. Puede definir todas las funciones auxiliares que crea conveniente.

Ayuda: El predicado `integer?` puede ser útil.

3 Estructuras

Ejercicio 3.

1. Podemos representar el stock de un producto en el supermercado mediante una estructura con tres campos:

   - 1er campo: el nombre del producto
   - 2do campo: el precio unitario del producto
   - 3er campo: la cantidad de unidades disponibles del mismo.

   Diseñe una estructura `producto` que permita representar esta información.

2. Diseñe una función `monto-en-stock` que dado un elemento de tipo `producto` determine el monto que representan todas las unidades disponibles del mismo.

   Por ejemplo,

   ```
   (check-expect (monto-en-stock (make-producto "leche" 24.5 6)) 147)
   (check-expect (monto-en-stock (make-producto "jabón" 30 2)) 60)
   ```

   Incluya otros ejemplos apropiados en su diseño.

3. Diseñe una función `actualizar-stock` que dados dos elementos de tipo producto, se comporte como sigue:

   - Si tienen el mismo nombre de producto, debe devolver una nueva estructura donde el precio unitario es el promedio de los precios unitarios de cada uno; y la cantidad es la suma de ambas cantidades.
   - Si no tienen el mismo nombre, devuelve un String indicando tal situación.

   Por ejemplo:

   ```
   (check-expect (actualizar-stock ((make-producto "leche" 20 5) (make-producto "leche" 30 2)))
              (make-producto "leche" 25 7))
   ```

   Incluya otros ejemplos relevantes en su diseño.