Guía de Trabajos Prácticos VI – Programación Funcional 3

1. Escriba la función factorial.
2. Escriba una función que devuelva el largo de una lista sin utilizar la función definida en Racket.
3. Escriba una función que cuente la cantidad de apariciones de un elemento en una lista. El primer parámetro será el elemento a buscar y el segundo la lista en la que se deber buscar.

   `(count-elem 3 '(1 2 3 4 5 4 3 2 1 ))  → 2`
4. Defina un procedimiento **subst** que reciba tres parámetros (dos valores y una lista) y devuelva la lista con todos los componentes que son iguales al primer parámetro reemplazados por el valor del segundo parámetro.

   (subst 'c 'k '(c o c o n u t)) → (k o k o n u t)
5. Se desea crear un función que reciba como parámetros una lísta de átomos compuesto únicamente de letras y devuelva una lista agrupando los que son iguales en sublistas.

   Ej: (agrupar '(A A B C A B A D C)) --> ((AAAA) (BB) (CC) (D))
6. Escriba en Racket el procedimiento (concatenar l1 l2) que recibe dos listas l1 y l2 como argumento y retorna una única lista que contiene primero todos los elementos de l1 seguido de todos los elementos de l2 (no puede usar el procedimiento interno append que viene en algunas implementaciones de Racket).
7. Defina en Racket un procedimiento recursivo que encuentre el primer elemento de una lista que es un número. Debe retornar el número si lo encuentra, sino retornar null.
   Ej: **`(primer-num '((1 . 2) 'a (b) (5) 6 8 'a 9) → 6`**
8. Defina un procedimiento en Racket llamado **attach-at-end** que reciba como parámetro un valor y una lista y retorne la lista con los mismos valores excepto el que se pasó por parámetro que se agregará al final.
   `(attach-at-end 'prueba '(esto es una)) → (esto es una prueba)`
9. Se desea crear un programa que permita convertir un lote de datos de un formato a otro. Los datos llegan en formato de lista de listas, donde el primer elemento determina el contenido de la lista y el segundo tiene la lista de datos. Los datos pueden venir en formato texto, decimal o booleano y se desea obtener una lista igual pero con todos sus componentes en formato decimal y todos positivos.
   Ej:
   ```
   (conv-datos '(
    ("D" (1 2 -3 4 -5))
    ("T" ("6" "7" "8"))
    ("B" ("V" "F"))
    )   -------------> '((1 2 3 4 5) (6 7 8) (1 0))
   ```
10. Defina un procedimiento en Racket que reciba como parámetros una lista de pares que representan puntos en el plano y devuelva una lista de las distancias entre cada uno de ellos tomados de a pares secuencialmente (distancia entre el primero y el segundo, luego entre el segundo y el tercero, y así sucesivamente). Utilice un solo Define.
    `(distancias '((1 . 1) (1 . 2) (2 . 2) (2 . 1) (1 . 1)) → (1 1 1 1)`

11. Cree una función llamada "group" que reciba como parámetros un lista propia y devuelva la misma agrupando por tipo elementos que contiene por tipo. Los grupos a considerar serán: números, string, booleanos, símbolos y otros. Deberán aparecer en ese órden.
Si de un grupo no hubiera elementos el mismo no debe aparecer en la lista resultado.

    Ejemplo:

    ```
    > (group '(A #t #f 3 2 4.5 "texto" "de" "prueba" #(1 2 3) S (a b c)))
    -->
    ((3 2 4.5) ("texto" "de" "prueba") (#t #f) (A S) (#(1 2 3) (a b c)))

    > (group '(A 3 2 4.5 "texto" "de" "prueba" S ))
    -->
    ((3 2 4.5) ("texto" "de" "prueba") (A S) )

    Utilice un solo define.
    ```

12. Cree las funciones to-base24 y from-base24 para convertir a, y desde base24. Los números en base 24 estarán formados por los caracteres: 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N. Para realizar el programa utilice vectores para referenciar los elementos correspondientes. Cada función deberá estar definida en un único define.

    ```
    Ej:
    (to-base24 555) --> N3
    (to-base24 550) --> MM
    (from-base24 'MM') --> 550
    ```