Lenguajes de Programación
Tarea Nº1

Profesor: Éric Tanter
Auxiliar: Ismael Figueroa

Fecha de Entrega: Lunes 21 de Marzo[1]
Para cada función recuerde escribir su contrato, descripción y tests

Parte I: Map/Reduce

En un conjunto de documentos de texto se desea obtener para cada uno de ellos la frecuencia con que aparece cada palabra. Además se requiere obtener un resumen total de las frecuencias de palabras en todos los documentos. Considere que un documento está representado como una lista de Scheme, cuyos elementos son strings.

1. (**1 pto.**) Implemente la función **summarize-document** que obtiene la frecuencia de palabras de un documento dado. Hint: Represente la frecuencia de palabras de un documento como una lista, ordenada por palabra, de pares. Su interfaz debe ser la siguiente: **(summarize-document doc)**. Ejemplo:

```
>(summarize-document '("hello" "world" "hello"))
(("hello" . 2) ("world" . 1))
>(summarize-document '())
()
```

2. (**1 pto.**) Implemente la función **merge-summaries** que combina los resultados de los resúmenes de dos documentos. Hint: Considere ordenar los documentos por palabra y luego comparar sus primeros elementos para realizar la combinación. Su interfaz debe ser la siguiente: **(merge-summaries s1 s2)**. Ejemplo:

```
>(merge-summaries '(("hello" . 2) ("world". 1)) '(("foo". 2) ("hello". 7)))
(("foo" . 2) ("hello" . 9) ("world" . 1))
>(merge-summaries '(("hello" . 5)) '())
(("hello". 5))
```

3. (**0.3 pto.**) Implemente la función **summarize-document-list** para obtener los resúmenes de una lista de documentos. Use **map** y **summarize-document**. Su interfaz debe ser: **(summarize-document-list document-list)**. Ejemplo:

```
>(summarize-document-list '(("hello" "world" "hello") ("foo" "bar")))
((("hello" . 2) ("world" . 1)) (("foo" . 1) ("bar" . 1)))
```

[1] Vea las reglas de entrega en http://pleiad.dcc.uchile.cl/teaching/cc4101

4. (**0.7 pto.**) Implemente la función **summarize-reduce** que al recibir una lista de resúmenes de documentos retorna el resumen global de todos los documentos. Use **foldr**/**foldl** y **summarize-merge**. Ejemplo:

```
>(summarize-reduce '( (("hello" . 2) ("world" . 1))
                      (("foo" . 1) ("bar" . 1) ("world" . 3))
                      (("world" . 1) ("wide" . 1) ("web" . 1)) ))
(("hello" . 2) ("world" . 5) ("foo" . 1) ("bar" . 1) ("wide" .1 ) ("web" . 1))
```

Parte II: Estructuras de Datos Recursivas

Considere el desarrollo de un administrador de recursos de un sistema de archivos que debe manipular archivos y carpetas. Tanto archivos como carpetas se identifican por un nombre. Además, las carpetas pueden contener una cantidad arbitraria de archivos y carpetas.

1. (**0.2 pto.**) Defina una gramática BNF de nombre **FSResource** para representar los recursos del sistema de archivos. Hint: considere que una carpeta tiene como elemento una lista de valores **FSResource**.

2. (**0.3 pto.**) Implemente usando define-type la gramática descrita anteriormente. Llame a este tipo **FSResource**.

3. (**2.5 pto.**) Implemente las funciones **file-copy** y **file-move**. Ambas reciben como parámetros el nombre del archivo (o carpeta) a copiar/mover y de la carpeta destino. Asuma que cada archivo y carpeta tiene un nombre único. Cada función debe retornar un elemento **FSResource** con la nueva organización de los recursos. Indique los errores correspondientes si el origen o destino son inexistentes. Las interfaces son: **(file-copy from-name to-name fs-resource)** y **(file-move from-name to-name fs-resource)**.