UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Tarea 5

INTEGRANTES

Torres Valencia Kevin Jair - 318331818
Aguilera Moreno Adrián - 421005200
Rivera Silva Marco Antonio - 318183583

PROFESORA

Karla Ramírez Pulido

AYUDANTES

Alan Alexis Martínez López
Manuel Ignacio Castillo López
Alejandra Cervera Taboada

ASIGNATURA

Lenguajes de Programación

6 de noviembre de 2022

1. Utiliza el paso de parámetros que se indica para evaluar la siguiente expresión.

```
{with{{a 8}
      {b -8}
      {swap{fun{x y}
            {with{{tmp x}}
              {seqn{set x y}
                   {set y tmp}}}}}}

 {seqn {swap a b}
      {-a{+b a}}}}
```

a. Paso de parámetros por valor.
Se tiene la representación del ambiente como:

swap	**{fun{x y}{with{{tmp x}}{seqn{set x y}{set y tmp}}}}**	0x12
b	-8	0x11
a	8	0x10

Evaluando {swap a b}, se tiene que:

{fun{x y}{with{{tmp x}}{seqn{set x y}{set y tmp}}} 8 -8}
{with{{tmp 8}}{seqn{set x -8}{set y tmp}}}
{seqn{set x -8}{set y 8}}
{set x -8} // Ahora $x = a = -8$
{set y 8} // Ahora $y = b = 8$

Donde sus parámetros son:

- Parámetro Formales: x y
- Parámetro Reales: $0x10$ $a = 8$ y $0x11$ $b = -8$

Ahora se puede evaluar {-a{+b a}}, se tiene que:
{- 8{+b 8}}
{- 8{+ (-8) 8}}
{- 8{0}} = 8

b. Paso de parámetros por referencia.
Se tiene la representación del ambiente como:

swap	**{fun{x y}{with{{tmp x}}{seqn{set x y}{set y tmp}}}}**	0x12
b	-8	0x11
a	8	0x10

Evaluando {swap a b}, se tiene que:

{fun{x y}{with{{tmp x}}{seqn{set x y}{set y tmp}}} 8 -8}
{with{{tmp 8}}{seqn{set x -8}{set y tmp}}}
{seqn{set x -8}{set y 8}}
{set x -8} // Ahora $x = a = -8$
{set y 8} // Ahora $y = b = 8$

Por lo que modificamos el ambiente, quedando como:

swap	**{fun{x y}{with{{tmp x}}{seqn{set x y}{set y tmp}}}}**	0x12
b	8	0x11
a	-8	0x10

Donde sus parámetros son:

- Parámetro Formales: x y
- Parámetro Reales: $0x10$ $a = -8$ y $0x11$ $b = 8$

Ahora se puede evaluar {-a{+b a}}, se tiene que:
{- (-8){+b (-8)}}
{- (-8){+ 8 (-8)}}
{- (-8){0}} = -8

2. Define la función recursiva ocurrencias que recibe dos listas y devuelve una lista de parejas, en donde cada pareja contiene en su parte izquierda un elemento de la segunda lista y en su parte derecha el número de veces que aparece dicho elemento en la primera lista. Por ejemplo:

```
>(ocurrencias '(2 6 8 6 2 1 2 2 0 3) '(2 6 9))
' ((2 . 4) (6 . 1) (9 . 0))
```

Solución. Para este ejercicio, damos la función recursiva que resuelve el problema dado y escrita en el lenguaje `Racket`. Esto es

```
; Funcion recursiva que cuenta las ocurrencias de list2 en list1:
(define (ocurrencias list1 list2)
  (define (ocurrencia list1 x) ; Auxiliar que cuenta ocurrencias de x en list1
    (if (empty? list1)
        0
        (if(equal? x (first list1))
           (+ 1 (ocurrencia (rest list1) x))
           (ocurrencia (rest list1) x)))
  )

  (if (empty? list2)
      empty
      (cons (cons (first list2) (ocurrencia list1 (first list2)))
            (ocurrencias list1 (rest list2))))
)
```

3. A partir del Ejercicio 2, muestra los registros de activación generados por la función con la siguiente llamada.

```
(ocurrencias '(1 2 3) '(1 2))
```

Solución. A continuación se muestran los registros de activación por llamada recursiva, estos son

Registro principal:

Resultado: '((1 . 1) (ocurrencias '(1 2 3) '(2)))	0x28
...	
(ocurrencia '(1 2 3) 1)	0x13
Cuerpo/definición	0x12
'(1 2) '(1 2 3)	0x11
ocurrencias	0x10

Subregistro de activación 0x13

Resultado: 1	0x27
...	
(+ 1 (ocurrencia '(2 3) 1))	0x17
Cuerpo/definición	0x16
1 '(1 2 3)	0x15
ocurrencia	0x14

Subregistro de activación 0x17

Resultado: (ocurrencia '() 1)	0x23
Cuerpo/definición	0x22
1 '(3)	0x21
ocurrencia	0x20

Subregistro de activación 0x20

Resultado: (ocurrencia '(3) 1)	0x20
Cuerpo/definición	0x19
1 '(2 3)	0x18
ocurrencia	0x17

Subregistro de activación 0x23

Resultado: 0	0x26
Cuerpo/definición	0x25
1 '()	0x24
ocurrencia	0x23

Segundo registro principal de la función `ocurrencias`:

Resultado: '((2 . 1) (ocurrencias '(1 2 3) '()))	0x46
...	
(ocurrencia '(1 2 3) 2)	0x32
Cuerpo/definición	0x31
'(2) '(1 2 3)	0x30
ocurrencias	0x29

Subregistro de activación 0x33

Resultado: `(ocurrencia '(2 3) 2)`	0x36
Cuerpo/definición	0x35
2 `'(1 2 3)`	0x34
`ocurrencia`	0x33

Subregistro de activación 0x39

Resultado: `(ocurrencia '() 2)`	0x42
Cuerpo/definición	0x41
2 `'(3)`	0x40
`ocurrencia`	0x39

Subregistro de activación 0x36

Resultado: `(+ 1 (ocurrencia '(3) 2))`	0x39
Cuerpo/definición	0x38
2 `'(2 3)`	0x37
`ocurrencia`	0x36

Subregistro de activación 0x42

Resultado: `0`	0x45
Cuerpo/definición	0x44
2 `'()`	0x43
`ocurrencia`	0x42

Tercer registro principal de la función `ocurrencias`:

Resultado: `'()`	0x49
Cuerpo/definición	0x48
`'()` `'(1 2 3)`	0x47
`ocurrencias`	0x46

4. Usando recursión de cola optimiza la función del Ejercicio 2. Toda función auxiliar ocupada debe ser optimizada.

Solución. Para este ejercicio, damos la función recursiva de cola que resuelve el problema dado y escrita en el lenguaje `Racket`. Esto es

```
; Funcion recursiva de cola que cuenta las ocurrencias de list2 en list1:
(define (cola_ocurrencias list1 list2)
  (ocurrencias_cola list1 list2 '())
)

; Funcion recursiva de cola que cuenta las apariciones de x en list:
(define (ocurrencia_cola list x acc)
  (if (empty? list)
      acc
      (if (equal? x (first list))
          (ocurrencia_cola (rest list) x (+ acc 1))
          (ocurrencia_cola (rest list) x acc)))
)

; Funcion recursiva de cola que implementa una optimizacion para encontrar las
   ocurrencias de list2 en list1:
(define (ocurrencias_cola list1 list2 acc)
  (if (empty? list2)
      acc
      (ocurrencias_cola       ;Argumentos de en la llamada recursiva:
       list1 (rest list2)
       (cons acc (cons (first list2) (ocurrencia_cola list1 (first list2) 0))))))
```

5. A partir del Ejercicio 4, muestra los registros de activación generados por la función con la siguiente llamada.

```
(ocurrencias '(1 2 3) '(1 2))
```

Solución. Al igual que en el ejercicio 3, las llamadas recursivas son las mismas[1] exceptuando las modificaciones en el nombre de las respectivas funciones y la manera en la que se guarda el acumulador, acontinuación presentamos como se ve el último registro de activación al llama la función con los valores indicados, esto es

Resultado: '((() 1 . 1) 2 . 1)	0x50
Cuerpo/definición	0x51
'() '(1 2 3)	0x50
ocurrencias_cola	0x49

[1] 11 registros más la llamada a la función `cola_ocurrencias` que genera un registro extra, en total 12 registros.