Segundo examen opcional - Fundamentos de lenguajes de programación - Duración: 3 horas

Carlos Andres Delgado S, Msc *

09 de Septiembre 2019

1. (30 puntos) Dada la siguiente expresión en lenguaje visto en el curso, suponiendo como ambiente inicial el vacío

```
let
    f = proc(x y)
            begin
                set x = +(x,y);
                set y = -(x,y);
                +(x,y)
            end
    g = proc(x y)
           begin
               set x = *(x,y);
               set y = -(x,y);
               -(x,y)
           end
    x = 2
    y = 3
in
  let
     m = (f x y)
     h = (f x y)
     i = (g (f x y) (f x y))
     in
       let
         k = (f (g m h) (g m h))
       in
         +(m, +(h, i))
```

   Tenga en cuenta para este ejercicio que los rands se evalúan de izquierda a derecha.

   - (20 puntos) Dibuje los ambientes suponiendo paso por referencia.
   - (5 puntos) El valor total de la expresión en paso por referencia es 12055. Indique los valores finales de x,y,m,h,i,k.

2. (35 puntos) Dada la siguiente expresión:

```
let
    a = proc((int->(int->int)) b, ? c, int d) if (c d) then proc(? e) proc(? f) +(e,f) else b
    g = proc(? h) <(h,0)
    i = proc(? j) proc(? k) +(j,k)
    l = +(3,7)
in
    let
        m = (a i g l)
    in
        let
            n = proc(? o, ? p) if (p l) then (m (proc (? p) +(p,l) +(l,3))) else (m l)
        in
            (n a g)
```

*carlos.andres.delgado@correounivalle.edu.co

Importante: Este punto únicamente puede ser respondido en las tablas que aparecen a continuación, no se tomará en cuenta respuesta por fuera de ellas.

Variable	**Expresion**	**Tipo inferido**
T_a	a	
T_b	b	
T_c	c	
T_d	d	
T_e	e	
T_f	f	
T_g	g	
T_x	h	
T_y	i	
T_j	j	
T_k	k	
T_l	l	
T_m	m	
T_n	n	
T_o	o	
T_0	let ... in (n a g)	
T_1	if (c d) then proc(? e) proc(? f) +(e,f) else b	
T_2	(c d)	
T_3	proc(? e) proc(? f) +(e,f)	
T_4	proc(? f) +(e,f)	
T_5	+(e,f)	
T_6	<(h,0)	
T_7	proc(? k) +(j,k)	
T_8	+(j,k)	
T_9	+(3,7)	
T_{10}	proc(? o, ? p) ...	
T_{11}	if (p l) then ... else ...	
T_{12}	(p l)	
T_{13}	(m (proc (? p) +(p,l) +(l,3)))	
T_{14}	(proc (? p) +(p,l)	
T_{15}	+(p,l)	
T_{16}	+(l,3)	
T_{17}	(m l)	

A continuación escriba las ecuaciones que derivan de las siguientes expresiones:

Expresión	**Ecuaciones**
a = proc((int->(int->int)) b, ? c, int d) ...	
if (c d) then ... else ...	
(c d)	
proc(? e) proc(? f) +(e,f)	
proc(? f) +(e,f)	
+(e,f)	
g = proc(? h) ...	
<(h,0)	
i = proc(? j)...	
proc(? k) ...	
+(j,k)	
m = (a i g l)	
n = proc(? o, ? p) ...	
if (p l) then ... else ...	
(p l)	

Expresión	Ecuaciones
(m (proc (? p) +(p,l) +(l,3)))	
(proc (? p) +(p,l) +(l,3))	
proc (? p) +(p,l)	
+(l,3)	
(m l)	
(n a g)	

3. (35 puntos) **Parte práctica:** En los siguientes puntos se deben implementar los tipos en el interpretador de tipos por chequeo con las funciones type-of-expression y check-equal-type!. Si no se trabaja con estas funciones su implementación no será válida. Las reglas que agregue se deben ver reflejadas en el interpretador. Comente los cambios que ha realizado como justificación de su trabajo, si no realiza los comentarios no se valdrá el punto.

   - (10 puntos) Implemente el tipo string, el cual consiste de un identificador entre comillas simples '. Ejemplo:

```
let
   a = 'hola'
   in
      a
```

     Así mismo implemente la primitiva length que permite conocer el tamaño de un string. La regla a implementar de esta primitiva es $(string- > bool)$. Lo siguiente debería funcionar:

```
let
   a = 'hola'
   b = 'holaMundo'
   f = proc(string c) if >( (length(c), 5) then 3 else 4
   in
      let
         d = (f a)
         e = (f b)
      in
         +(d,e)
```

     La siguiente expresión debe ser 7, siendo que d es 3 y e es 4.

   - (25 puntos) Implemente el chequeo de listas. Las listas tienen la siguiente gramática

```
<expression> ::= [  <expression>* ]  (list-exp)
```

La regla de las listas es que todos sus elementos deben ser del mismo tipo así:

```
[ t1 t2 t3 ... tn]
t1 = t2 = t3 = ... = tn
```

El siguiente ejemplo debería funcionar sin problema:

```
let
    f = proc(int a) +(a, *(2,a))
    c = 3
    in
       [ (f c) +(4,3) c ]
```

El siguiente ejemplo debe indicar que la lista espera enteros y no booleanos.

```
let
    f = proc(int a) +(a, *(2,a))
    c = 3
    in
       [ (f c) +(4,3) true c 3]
```

El tipo de la lista se determina a partir del tipo de su primer elemento.

Entregue en el campus virtual el interpretador de chequeo de tipos modificado.