Primer examen opcional - Parte 2 60 % parcial
Fundamentos de lenguajes de programación
Duración: 2 horas

Carlos Andres Delgado S, Ing *

21 de Octubre 2015

Nombre:_____
Código:_____

Importante: Suba al campus virtual el interpretador modificado. Incluya comentarios en cada función que construya, en los comentarios deben ir casos de prueba.
Utilice el interpretador definido en clase que consta de evaluación de expresiones condicionales, procedimientos y procedimientos recursivos. Descargue del campus virtual el interpretador **Interpretador procedimiento recursivos**

1. Añadiendo primitivas [15 puntos]

Añada al lenguaje las primitivas de números enteros **equal?**, **greater?** y **less?**.

equal?(1,2) Retorna #f
greater?(2,1) Retorna #t
less?(2,1) Retorna #t

1. (7 puntos) Implemente y describa los cambios que deben realizarse a la especificación de la gramática y al tipo de dato primitive

2. (8 puntos) Implemente y describa los cambios que deben realizarse al procedimiento applyprimitive

2. Añadiendo operaciones al lenguaje [55 puntos]

1. (25 puntos) Añada las operaciones cons, car, cdr, null? y emptylist al lenguaje. Por ejemplo:

```
let x = 4
  in cons(x,
     cons(cons(-(x,1),
          emptylist),
     emptylist))
```

   Debe retornar: (4 (3)).

   Tomando la definición de **x** del ejemplo anterior, las instrucciones:

```
null?(x)
car(x)
cdr(x)
emptylist
```

   Retornan respectivamente **#f**, 4, ((3))y ()

2. (30 puntos) Extienda el lenguaje para añadir la operación **let***, por ejemplo:

```
let x = 30
  in let* x = -(x,1) y = -(x,2)
  in -(x,y)
```

   Debe retornar 2. La diferencia con **let** radica que cuando se declara y se utiliza el valor de x declarado inmediatamente antes, en este caso **y** toma el valor de 27 y no 28 como lo haria en el **let** usado en el curso.

3. Añadiendo arboles binarios de búsqueda al interpretador [30 puntos]

En este ejercicio se le propone extender el conjunto de valores expresados del lenguaje para manejar arboles binarios de búsqueda.

Los arboles binarios de busqueda tienen las siguientes propiedades

1. Todo nodo tiene asociado un valor entero x

2. Dado un nodo cualquiera con un valor x, el subarbol hijo izquierdo contiene valores **menores o iguales a x**

3. Dado un nodo cualquiera con un valor x, el subarbol hijo derecho contiene valores **mayores o iguales a x**

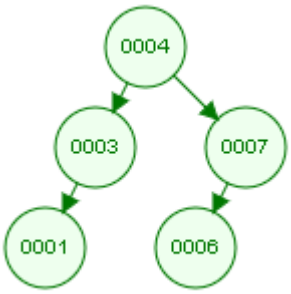
1. (15 puntos) Implemente la creación de arboles binarios de busqueda **create-bintree(list)**.

   Por ejemplo:

```
create-bintree('(4 3 1 7 6))
```

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

Los elementos se van insertando al arbol siguiente las reglas en el orden de la lista por lo que debe crear el siguiente árbol:



Indique la BNF para arboles binarios de búsqueda y los cambios que deben realizarse en la gramática del interpretador. Es importante aclarar que en este caso se va a generar sólo el árbol de sintaxis abstracta del arbol binario de búsqueda.

2. (15 puntos) Implemente la función **viewtree(arbolbinario)** la cual recibe un arbol binario de busqueda y retorna una representación en listas **(valor hijoizquierdo hijoderecho)**, por ejemplo:

```
1 miarbol = create-bintree('(4 3 1 7 6))
2 viewtree(miarbol)
3 ( 4
4   (3 (1 emptybintree emptybintree) emptybintree)
5   (7 (6 emprybintree emptybintree) emptybintree)
6 )
```