1) Diseñe un interpretador para la siguiente gramática que realiza operaciones con notación infija:

Valores denotados: Texto + Número + Booleano + ProcVal

Valores expresado: Texto + Número + Booleano + ProcVal

<programa> :=  <expresion>

               un-programa (exp)

<expresion> := <numero>

               numero-lit (num)

            := "\""<texto> "\""

               texto-lit (txt)

            := <identificador>

               var-exp (id)

             := (expresion <primitiva-binaria> expresion)

               primapp-bin-exp (exp1 exp2)

              := <primitiva-unaria> (expresion)

               primapp-un-exp (exp)

<primitiva-binaria> :=  + (primitiva-suma)

  :=  ~ (primitiva-resta)

  :=  / (primitiva-div)

  :=  \* (primitiva-multi)

  :=  concat (primitiva-concat)

<primitiva-unaria>:=  longitud (primitiva-longitud)

            :=  add1 (primitiva-add1)

            :=  sub1 (primitiva-sub1)

Tenga en cuenta que:

<numero>: Debe definirse para valores decimales y enteros (positivos y negativos)

<texto>: Debe definirse para cualquier texto escrito en racket

<identificador>: En este lenguaje todo identificador iniciará con el símbolo  @, es decir las variables @x y @z son válidas

2)Defina un ambiente inicial con las variables (@a @b @c @d @e) con valores (1 2 3 "hola" "FLP") y modifique su función evaluar-expresión para que acepte dicho ambiente.

-Diseñe una función llamada (buscar-variable) que recibe un símbolo (identificador) y un ambiente, retorna el valor si encuentra la variable en el ambiente. En caso contrario: "Error, la variable no existe"

Pruebe:

--> @a

1

--> @b

2

--> @e

"FLP"

3) Implemente los Booleanos:

En una expresión numérica, 0 es falso, cualquier otro caso es verdadero. Para esto diseñe la función valor-verdad? que realiza esta verificación.

4) Extienda la gramática con condicionales:

<expresion> := Si <expresion> entonces <expresion>  sino <expresion> finSI

               condicional-exp (test-exp true-exp false-exp)

Debe probar:

--> Si (2+3) entonces 2 sino 3 finSI

2

--> Si (length(@d) ~ 4) entonces 2 sino 3 finSI

3

5) Implemente declaración de variables locales:

<expresion> := declarar (<identificador> = <expresion> (;)) { <expresion> }

               variableLocal-exp (ids exps cuerpo)

Debe probar:

--> declarar (@x=2;@y=3;@a=7){

       (@a+(@x~@y))

    }

6

--> declarar (@x=2;@y=3;@a=7) {

      (@a+@b)

   }

9

6) Extienda la gramática para crear procedimientos

<expresion> := procedimiento (<identificador>\*',') haga <expresion> finProc

              procedimiento-ex (ids cuero)

Para esto debe definir un datatype para la cerradura (o ProcVal) que debe tener 3 campos:

1. Lista ID del procedimiento

2. Cuerpo del procedimiento

3. Ambiente donde fue declarado

(define-datatype procVal procVal?

  (cerradura

   (lista-ID (list-of symbol?))

   (exp expresion?)

   (amb ambiente?)

   )

  )

Debe probar:

--> procedimiento (@x,@y,@z) haga ((@x+@y)+@z) finProc

#(struct:cerradura (@x @y @z) #(struct:primapp-bin-exp #(struct:primapp-bin-exp #(struct:var-exp @x) #(struct:primitiva-sum) #(struct:var-exp @y)) #(struct:primitiva-sum) #(struct:var-exp @z)) #(struct:extendido (@a @b @c @d @e) (1 2 3 "hola" "FLP") #(struct:vacio)))

Se debe retornar una cerradura

7) Extienda la gramática para evaluar procedimientos:

<expresion> :=  "evaluar" expresion enviando  (expresion ",")\*  finEval

Debe probar:

-->  declarar (

      @x=2;

      @y=3;

      @a=procedimiento (@x,@y,@z) haga ((@x+@y)+@z) finProc

     ) {

         evaluar @a (1,2,@x) finEval

       }

5

--> declarar (

     @x=procedimiento (@a,@b) haga ((@a\*@a) + (@b\*@b)) finProc;

     @y=procedimiento (@x,@y) haga (@x+@y) finProc

    ) {

      ( evaluar @x(1,2) finEval + evaluar @y(2,3) finEval )

     }

10

--> declarar (

      @x= Si (@a\*@b) entonces (@d concat @e) sino length((@d concat @e)) finSI;

      @y=procedimiento (@x,@y) haga (@x+@y) finProc

   ) {

      ( length(@x) \* evaluar @y(2,3) finEval )

     }

35

8) Extienda la gramática para incluir llamados recursivos. Proponga una definición en la gramática e impleméntela.

9) Dibuje el árbol de sintaxis abstracta de los ejercicios del punto 7.

10) Dibuje el diagrama de ambientes de los ejercicios del punto 7 (tenga en cuenta el ambiente inicial del punto 1).

11) Utilización del lenguaje de programación:

a) Escriba un programa en su lenguaje de programación que contenga un procedimiento que permita calcular el ·rea de un circulo dado un radio (A=PI\*r\*r). Debe incluir valores flotantes en su lenguaje de programación.

b) Escriba un programa en su lenguaje de programación que contenga un procedimiento que permita calcular el factorial de un número n.

c) Escriba un programa en su lenguaje de programación que contenga un procedimiento que permita calcular una multiplicación de forma recursiva a través de sumas. Por ejemplo: la multiplicación 3\*4 equivale a 4+4+4.

d) Escriba un programa recurso en su lenguaje de programación que permita sumar, restar, multiplicar dos números haciendo uso solamente de las primitivas add1 y sub1

**Aclaraciones**

1. El taller es en grupos de ma ́ximo tres (3) alumnos.
2. La solución del taller debe ser subida al campus virtual a más tardar el día 7 de marzo, 2020 (23:59). Se debe subir al campus virtual en el enlace correspondiente a este taller un archivo comprimido .zip que siga la convención CódigodeEstudiante1-CódigodeEstudiante2- CódigodeEstudiante3-Taller3FLP20192.zip. Este archivo debe contener el archivos: interpretador.rkt.
3. En las primeras líneas del interpretador deben estar comentados los nombres y los códigos de los estudiantes participantes.
4. Se debe incluir para cada procedimiento un comentario que explique lo que realiza  y para qué es empleada.

**Entregas Tardías o por Otros Medios**

1. Este taller sólo se recibirá a través del campus virtual. Adicionalmente, sólo se evaluarán los documentos solicitados en el punto 2 de la sección anterior. Cualquier otro tipo de correo o nota aclaratoria será descartado. Sólo se aceptan envíos por fuera del horario establecido bajo excusas de fuerza mayor validadas a través de la dirección de la Escuela.

2. Las entregas tarde serán penalizadas así: (-1pt) por cada hora de retraso o fracción. Por ejemplo, si usted realiza su entrega y el campus registra las 12:00 am (i.e., 1 seg después de la hora de entrega), usted está incurriendo en la primer hora de retraso. Asegurese con mínimo dos horas de anticipación que el link de carga funciona correctamente toda vez que es posible incurrir en una entrega tardía debido a los tiempos de respuesta.