LENGUAJES DE PROGRAMACIÓN

Semestre Enero-Mayo de 2020

TAREA 2

Forma de trabajo: En equipos de 2 integrantes
Forma de entrega: Como asignación en Canvas
Fecha de entrega: viernes 28 de febrero de 2020

Esta tarea tiene como objetivo el que pongas en práctica los conocimientos adquiridos en la clase de Matemáticas Computacionales y reforzados en nuestro curso, al guiarte en la implementación de un scanner y un parser de un mini lenguaje.

ESPECIFICACIÓN DEL LENGUAJE:

El lenguaje a analizar léxica y sintácticamente para esta tarea es super pequeño basado en la notación funcional del Cálculo Lambda. Este mini lenguaje es especificado mediante la siguiente gramática libre de contexto en formato BNF:

```
<P> ::= <E> $
<E> ::= var | num | op <E> | <E><E> | \var.<E> | (<E>)
```

Donde los símbolos no terminales aparecen entre ángulos **<P>** y **<E>**, haciendo referencia a el programa y las expresiones, respectivamente. O sea, un programa es una expresión sucedida por el símbolo terminal $, mientras que hay 6 formas para construir una expresión. Los símbolos terminales incluyen los caracteres especiales: fin del programa ($), los paréntesis, el punto y la diagonal invertida (\); también incluyen los caracteres para incluir variables (**var**), números enteros (**num** = uno o más dígitos) y operadores (**op**), tal y como se muestra abajo.

```
var   ε  {u, v, w, x, y, z}
num   ε  [0-9]+
op    ε  {+, -, *, /}
```

Los símbolos terminales deberán ser identificados y regresados por el analizador léxico (scanner) del lenguaje, mientras que el cumplimiento de la gramática deberá ser verificado por el analizador sintáctico (parser).

A partir del léxico y la sintaxis establecidos se pueden reconocer las palabras y expresiones bien construidas, como los ejemplos correctos de abajo. Pero también se deben de detectar errores en las palabras y expresiones erróneas como los ejemplos incorrectos de más abajo.

Ejemplos correctos:

```
x $
34 $
+ y 2 $
(\z.* z 15) 21 $
(\x.\y./ x y)(+ u 3) 1 $
```

Ejemplos incorrectos:

```
a $
3.4 $
y 2 + $
(/z.* z 15) 21 $
(\x.\y.\ x y)(+ u 3) 1 $
```

PROBLEMA 1

a) Diseñen y dibujen el diagrama de estados (autómata) que servirá para reconocer los elementos del léxico de este lenguaje.
b) Modifiquen la gramática libre de contexto (BNF) del lenguaje, de forma que pueda ser analizada por un analizador de sintaxis recursivo descendente, como el visto en clase. Para esto tienes que eliminar ambigüedades generadas por reglas recursivas por la izquierda y reglas con factores comunes por la izquierda.

PROBLEMA 2

Implementen un programa en **Python** que implemente el scanner (siguiendo la técnica de la matriz de transiciones) y el parser (usando el método de descenso recursivo) que analice las oraciones del lenguaje. Este programa lo deberán construir mediante una adaptación del programa visto en clase. La interface de ejecución del programa es la siguiente:

- El programa deberá leer las oraciones desde el teclado.
- El programa realizará el análisis léxico y sintáctico sobre la entrada, mostrando en pantalla los lexemas (texto fuente) de los elementos léxicos que va reconociendo mientras no haya un error.
- Si el analizador detecta un error de léxico, deberá marcar el error en pantalla con el letrero ">>ERROR LÉXICO<<" y abortar la ejecución del programa.
- Si detecta un error de sintaxis, deberá marcar el error en pantalla con el letrero ">>ERROR SINTÁCTICO<<" y abortar la ejecución del programa.
- Si no se detectan errores, al terminar el análisis de la entrada, se deberá desplegar un letrero ">>ENTRADA CORRECTA<<".

Documentación a entregar en el archivo compreso M_Tarea2.zip, donde deberán sustituir la M por sus matrículas separadas por guiones bajos (_):

1. Un documento (.pdf) con el diagrama de estados utilizado para el análisis léxico y la gramática libre de contexto recursiva descendente (no ambigua) que fue utilizada para el análisis sintáctico.
2. El código fuente en el lenguaje Python que implementa la solución en software. Debe de incluir el archivo en Python (.py) con el código del escáner y otro con el código del parseador.

***NOTA**: Cualquier duda o situación no considerada o descrita en este documento, deberá ser consultada con el profesor.*

Forma de Evaluación

- **(20 puntos)** Respuestas al problema 1: incluye todo lo que se pidió y está bien hecho.
- **(70 puntos)** Programa de análisis léxico y sintáctico: el programa se ejecuta correctamente e identifica correctamente los distintos tipos de errores.
- **(10 puntos)** Datos de los integrantes del equipo, limpieza y claridad de la respuesta al problema 1, código con comentarios internos en el programa del problema 2.