APELLIDO: …… NOMBRE: ……DNI…… Lenguajes y Compiladores – UNLAM PRIMERA EVALUACION DE APRENDIZAJE	**20/05/2020**

Evaluación de Aprendizaje Nº 1
Unidades Temáticas: 1 a 8

Objetivo: Evaluar los aprendizajes sobre los contenidos de las unidades 1 a 8.

Alcance: Expresiones Regulares, Analizador Lexicográfico, Gramáticas (GLC y BNF), Analizador Sintáctico, Análisis Sintáctico (Parsing Ascendente), Errores léxicos y sintácticos

Evaluación: Deberá tener 4 ejercicios bien. De lo contrario se pedirá la reentrega.

Presentación: Las respuestas deben presentarse en un archivo formato PDF con el nombre *EA_1_Rtas_ApellidoyNombre*, en la Plataforma Miel el día miércoles 27/5/2020 a la hora 12 a.m.. Cada alumno deberá presentarlo a su profesor asignado en el archivo Profesores.

Sea el siguiente formato para una instrucción del tipo INSERT INTO de SQL:

INSERT INTO NombrePropietario. NombreTabla (campo$_1$, campo$_2$, ... campo$_n$) VALUES (valor$_1$, valor$_2$, …, valor$_n$)

Donde:

NombrePropietario es un identificador que comienza con letra y continua con letras ó dígitos y guiones bajos, pero no puede finalizar con estos últimos.

NombreTabla y campo$_i$ es un identificador que comienza con letra y continua con letras ó dígitos.

valor$_i$ puede ser un identificador (como los del tipo campo$_i$), o un identificador (como los del tipo campo$_i$) con el símbolo @ delante Ej. variable1, @variable2.

INSERT INTO y VALUES son palabras reservadas

Los **paréntesis** también forman parte de la instrucción.

Se pide:

Ejercicio Nro. 1

Definir los tokens necesarios a través de expresiones regulares para la instrucción INSERT INTO.

Ejercicio Nro. 2

Definir una gramática en formato BNF para la sentencia anterior utilizando los tokens definidos en el ejercicio 1.

APELLIDO: …… NOMBRE: ……DNI…… Lenguajes y Compiladores – UNLAM PRIMERA EVALUACION DE APRENDIZAJE	20/05/2020

Ejercicio Nro. 3

Sea la siguiente gramática que representa la sintaxis de una sentencia TAKE que se asigna a un identificador

id = TAKE (Operador ; cte ; [lista de identificadores]).

Esta función toma como entrada un *operador* de suma y multiplicación, una *constante* entera y una *lista de identificadores*. Esta función devuelve el valor que resulta de aplicar el *operador* a los primeros "n" elementos de la lista. El valor de n quedará establecido en la componente *cte*.
Los elementos de la lista están separados por blancos.
El resultado de la función puede ser utilizado en otras expresiones dentro del lenguaje.
En todo momento deberá validarse la cantidad definida en *cte* con la cantidad de elementos de la lista.
Ej : TAKE (+;1;[id]) Si id tomase el valor 3 devuelve 3

TAKE (+;3; [a b c d e]) Si a tomase el valor 10,b el valor 20, c el valor 30, d el 40 y e el valor 50) devuelve 60 (resultado de sumar los 3 primeros ids de la lista)

Gramática < {Asig, O, Lista} , { cte, id, asigna, take,[,],+,*,(,), ; } , S , Reglas

$S \rightarrow Asig$

$Asig \rightarrow id\ asigna\ take\ (\ O\ ;\ cte\ ;\ [Lista]\ )$

$Lista \rightarrow id$

$Lista \rightarrow Lista\ id$

$O \rightarrow +$

$O \rightarrow *$

Se pide:

a) Hacer la tabla SLR de parsing ascendente (Respetar el siguiente esquema de columnas)

id	(	)	asigna	take	+	*	cte	;	[	]	$	Asig	Lista	O

b) Probar el parsing para la cadena a = TAKE (*; 3; [a b c])

Ejercicio Nro. 4

Señale dos casos de error de compilación y márquelos en la tabla de parsing.

Ejercicio Nro. 5

¿Es posible que si esta gramática fuese ambigua genere conflictos en el armado de la tabla de parsing?