UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Integrantes:
Adrián Aguilera Moreno
Sebastián Alejandro Gutierrez Medina

Compiladores

Tarea 02

1. Considera la siguiente gramática:

$$\begin{array}{rcl} E & \to & -E \mid (E) \mid VE' \\ E' & \to & -E \mid \varepsilon \\ V & \to & \texttt{id}V' \\ V' & \to & (E) \mid \varepsilon \end{array}$$

1. Construye la tabla de parsing para un parser tipo LL(1) usando el cálculo de los conjuntos First y Follow que obtuviste en la tarea anterior.

2. Muestra lo que se obtiene al ejecutar el algoritmo para procesar la cadena $-\texttt{id}(-\texttt{id}) - \texttt{id}$. Incluye una tabla para ver el progreso del algoritmo donde se muestre el avance del procesamiento de la cadena y la evolución de la pila del parser.

2. Considera el siguiente fragmento de una gramática que abstrae el comportamiento de expresiones del lenguaje C:

$$E \to *E \mid \&E \mid E = E \mid E -> E \mid id$$

Esta gramática es ambigua pero se puede transformar en una no-ambigua usando la precedencia de operadores. En particular, el acceso a campos de una estructura, $E -> E$, tiene mayor precedencia que la derrefenciación y las expresiones para direcciones; además, estas tres tienen mayor precedencia que las asignaciones [1]

1. Escribe una gramática equivalente tipo **LL(1)** que incluya la precedencia descrita, muestra el proceso o describe las técnicas que uses para obtener esta nueva gramática.

2. Muestra la tabla de parsing para la gramática del inciso anterior.

3. Procesa la expresión `* * a -> b -> c = & * d` usando el algoritmo para **LL** y mostrando los estados del parser.

[1] Puedes revisar la tabla disponible en esta página para consultar la precedencia y asociatividad de los operadores en C.