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.

$$\begin{array}{rcl} (0)S & \to & E\$ \\ (1)E & \to & -E \\ (2)E & \to & (E) \\ (3)E & \to & VE' \\ (4)E' & \to & -E \\ (5)E' & \to & \varepsilon \\ (6)V & \to & \texttt{id}V' \\ (7)V' & \to & (E) \\ (8)V' & \to & \varepsilon \end{array}$$

Table 1: Tabla de Parsing

	-	(	)	id	$
S	(0)	(0)		(0)	
E	(1)	(2)		(3)	
E'	(4)		(5)		(5)
V				(6)	
V'	(8)	(7)	(8)		(8)

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.

Table 2: Tabla del progreso del algoritmo

Pila del Parser	Cadena	Regla aplicada
\$S	$-\texttt{id}(-\texttt{id})-\texttt{id}\$$	
\$E	$-\texttt{id}(-\texttt{id})-\texttt{id}\$$	$S \to E$
\$E-	$-\texttt{id}(-\texttt{id})-\texttt{id}\$$	$E \to -E$
\$E	$\texttt{id}(-\texttt{id})-\texttt{id}\$$	
\$E'V	$\texttt{id}(-\texttt{id})-\texttt{id}\$$	$E \to VE'$
\$E'V'id	$\texttt{id}(-\texttt{id})-\texttt{id}\$$	$V \to idV'$
\$E'V'	$(-\texttt{id})-\texttt{id}\$$	
\$E')E(	$(-\texttt{id})-\texttt{id}\$$	$V' \to (E)$
\$E')E	$-\texttt{id})-\texttt{id}\$$	
\$E')E-	$-\texttt{id})-\texttt{id}\$$	$E \to -E$
\$E')E	$\texttt{id})-\texttt{id}\$$	
\$E')E'V	$\texttt{id})-\texttt{id}\$$	$E \to VE'$
\$E')E'V'id	$\texttt{id})-\texttt{id}\$$	$V \to idV'$
\$E')E'V'	$)-\texttt{id}\$$	
\$E')E'	$)-\texttt{id}\$$	$V' \to \epsilon$
\$E')	$)-\texttt{id}\$$	$E' \to \epsilon$
\$E'	$-\texttt{id}\$$	
\$E-	$-\texttt{id}\$$	$E' \to -E$
\$E	$\texttt{id}\$$	
\$E'V	$\texttt{id}\$$	$E \to VE'$
\$E'V'id	$\texttt{id}\$$	$V \to idV'$
\$E'V'	\$	
\$E'	\$	$V' \to \epsilon$
\$	\$	$E' \to \epsilon$
\$	\$	ACEPTADA

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.

   Solución: A continuación se muestra la gramática con precedencia (solo un reacomodo) y con ambigüedad:

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

   quitando ambigüedades de derecha a izquierda tenemos que

   $$\begin{aligned} E &\to E -> P \mid P \\ P &\to *P \mid \&P \mid P = P \mid id \end{aligned}$$

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

ahora, quitemos las ambigüedades de izquierda a derecha, esto es

$$\begin{aligned} E &\rightarrow E -> P \mid P \\ P &\rightarrow *A \mid \&A \mid A = P \mid A \\ A &\rightarrow id \end{aligned}$$

lo anterio con base a Eliminación de ambigüedad.

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.