UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Integrantes:
Adrián Aguilera Moreno
Sebastián Alejandro Gutierrez Medina

Compiladores

Tarea 05

1. (2pts.) Demuestre que la siguiente gramática pertenece a la clase **LALR** pero no a la clase **SLR**.

$$E \to Aa \mid bAc \mid dc \mid bda \qquad A \to d$$

(1pt.) Además analice la cadena *bdc* mostrando la secuencia de acciones del parser.

2. (2.5pts.) La siguiente gramática genera expresiones en notación polaca inversa, es decir los argumentos preceden al operador:

$$E \to E\ E\ op \mid id \qquad\qquad op \to + \mid - \mid * \mid /$$

Suponer que cada *id* (identificadores en mayúsculas) tiene un atributo sintético `name` que es una cadena y los símbolos E y *op* tienen un atributo `val` que también es una cadena.
Diseña una gramática con atributos para organizar el atributo `val` de la raíz del parse tree para guardar la traducción de la expresión en notación infija (utiliza los paréntesis necesarios). Explica la idea que usas para definir las funciones semánticas.
Por ejemplo, si las hojas del parse tree (de izquierda a derecha) son $A\ A\ B\ -\ *\ C\ /$ entonces la raíz debe tener como atributo `val` la cadena $((A * (A - B))/C)$.

3. (2.5pts.) Extender la siguiente gramática con atributos para la regla $E \rightarrow E_1 * E_2$ y obtener el código de tres direcciones para la expresión $x = a[i] * b[j]$ donde a y b son arreglos de tamaño 2×3 y 2×2 respectivamente y cada uno de ellos almacena enteros cuyo tamaño es 4.

$S \rightarrow$ **id** = E ; { $gen($ $top.get($**id**$.lexeme)$ $'='$ $E.addr)$; }

| L = E ; { $gen(L.array.base$ $'['$ $L.addr$ $']'$ $'='$ $E.addr)$; }

$E \rightarrow E_1$ + E_2 { $E.addr =$ **new** $Temp()$;
$gen(E.addr$ $'='$ $E_1.addr$ $'+'$ $E_2.addr)$; }

| **id** { $E.addr = top.get($**id**$.lexeme)$; }

| L { $E.addr =$ **new** $Temp()$;
$gen(E.addr$ $'='$ $L.array.base$ $'['$ $L.addr$ $']')$; }

$L \rightarrow$ **id** [E] { $L.array = top.get($**id**$.lexeme)$;
$L.type = L.array.type.elem$;
$L.addr =$ **new** $Temp()$;
$gen(L.addr$ $'='$ $E.addr$ $'*'$ $L.type.width)$; }

$L \rightarrow$
L_1 [E] { $L.array = L_1.array$;
$L.type = L_1.type.elem$;
$t =$ **new** $Temp()$;
$L.addr =$ **new** $Temp()$;
$gen(t$ $'='$ $E.addr$ $'*'$ $L.type.width)$;
$gen(L.addr$ $'='$ $L_1.addr$ $'+'$ $t)$; }

4. (2pts.) Considera el siguiente fragmento de código:

```
if ( c[i] != 0 )
then
   a[i] := b[i] / c[i];
else
   a[i] := b[i];
```

Obtener las representaciones intermedias correspondientes a 1) árbol de sintaxis abstracta; 2) gráfica de control de flujo y 3) código de tres direcciones. Explica tus respuestas.
Discutir (ampliamente) las ventajas que consideras para cada representación.

5. (Hasta 1.5pt extra). Explica lo que es una gráfica de control de flujo. ¿Quién fue Frances Allen?