SOLUCIONES

Parte I: PREGUNTAS TIPO TEST. 30 %.

1. b)
2. c)
3. c)
4. b)
5. a)
6. a)
7. a)
8. b)
9. b)
10. c)
11. b)
12. c)
13. c)
14. b)
15. b)

Parte II: PREGUNTAS CORTAS. 10 %.

La gramática es:

$$
\begin{array}{lcl}
A & \rightarrow & x \\
 & | & (\ B\ ) \\
B & \rightarrow & A\ C \\
C & \rightarrow & +\ A\ C \\
 & | & \lambda
\end{array}
$$

Una gramática recursiva por la izquierda podría provocar una secuencia infinita de llamadas recursivas al procedimiento de un mismo no terminal.

Parte III: PROBLEMA. 60 %

Apartado 1. La gramática no es LL(1), porque es recursiva por la izquierda en los no terminales A y L. Para eliminar la recursividad izquierda debemos transformar previamente la gramática para que sea propia. Comenzamos aplicando el algoritmo que elimina las λ-producciones:

1. El conjunto de variables anulables es $V_{an} = \{D\}$;
2. Se elimina la regla $D \to \lambda$;
3. La nueva gramática se obtiene modificando las reglas de producción en las que aparece D:

$$\begin{array}{lcl} A & \to & A\ S \mid S \\ S & \to & \textbf{num} : D\ ; \mid \textbf{num} : ; \\ D & \to & L \\ L & \to & L\ \&\ B \mid B \\ B & \to & \textbf{id} > \textbf{num} \end{array}$$

La gramática obtenida es propia, puesto que es λ-libre, libre de ciclos y no tiene símbolos inútiles. Ahora aplicamos el algoritmo de eliminación de la recursividad por la izquierda de toda la gramática:

1. El orden de los no terminales es $V_N = \{A, S, D, L, B\}$;
2. Eliminación de la recursión inmediata en A:

$$\begin{array}{lcl} A & \to & S \mid SA' \\ A' & \to & S \mid SA' \end{array}$$

   Estas dos reglas se pueden simplificar en una: $A \to S \mid SA$;

3. Eliminación de la recursión inmediata en L:

$$\begin{array}{lcl} L & \to & B \mid BL' \\ L' & \to & \&\ B \mid \&\ BL' \end{array}$$

Por tanto, la gramática obtenida tras la eliminación de la recursividad izquierda es:

$$\begin{array}{lcl} A & \to & S \mid SA \\ S & \to & \textbf{num} : D\ ; \mid \textbf{num} : ; \\ D & \to & L \\ L & \to & B \mid BL' \\ L' & \to & \&\ B \mid \&\ BL' \\ B & \to & \textbf{id} > \textbf{num} \end{array}$$

Esta gramática se puede simplificar, eliminando el no terminal L y renombrando L':

$$\begin{array}{lcl} A & \to & S \mid SA \\ S & \to & \textbf{num} : D\ ; \mid \textbf{num} : ; \\ D & \to & B \mid BL \\ L & \to & \&\ B \mid \&\ BL \\ B & \to & \textbf{id} > \textbf{num} \end{array}$$

El último paso necesario antes de aplicar el análisis LL(1) es factorizar la gramática:

$$\begin{array}{lcl} A & \to & SA' \\ A' & \to & A \mid \lambda \\ S & \to & \textbf{num} : S' \\ S' & \to & D\ ; \mid ; \\ D & \to & BL' \\ L & \to & \&\ BL' \\ L' & \to & L \mid \lambda \\ B & \to & \textbf{id} > \textbf{num} \end{array}$$

Se puede comprobar que esta gramática sí es LL(1).

Apartado 2. La colección LR(1) está formada por los siguientes conjuntos de ítems:

$$
\begin{aligned}
I_0 = \{\ & [\ A' \to \cdot A\ ,\ \$\ ] \\
& [\ A \to \cdot AS\ ,\ \$/\mathbf{num}\ ] \\
& [\ A \to \cdot S\ ,\ \$/\mathbf{num}\ ] \\
& [\ S \to \cdot\mathbf{num}\ {:}D;\ ,\ \$/\mathbf{num}\ ]\ \}
\end{aligned}
$$

$$
\begin{aligned}
I_1 = \textsc{Goto}(I_0, A) = \{\ & [\ A' \to A\cdot\ ,\ \$\ ] \\
& [\ A \to A \cdot S\ ,\ \$/\mathbf{num}\ ] \\
& [\ S \to \cdot\mathbf{num}\ {:}D;\ ,\ \$/\mathbf{num}\ ]\ \}
\end{aligned}
$$

$I_2 = \textsc{Goto}(I_0, S) = \{\ [\ A \to S\cdot\ ,\ \$/\mathbf{num}\ ]\ \}$

$I_3 = \textsc{Goto}(I_0, \mathbf{num}) = \{\ [\ S \to \mathbf{num} \cdot {:}D;\ ,\ \$/\mathbf{num}\ ]\ \}$

$I_4 = \textsc{Goto}(I_1, S) = \{\ [\ A \to AS\cdot\ ,\ \$/\mathbf{num}\ ]\}$

$\textsc{Goto}(I_1, \mathbf{num}) = I_3$

$$
\begin{aligned}
I_5 = \textsc{Goto}(I_3, :) = \{\ & [\ S \to \mathbf{num} : \cdot\ D;\ ,\ \$/\mathbf{num}\ ] \\
& [\ D \to \cdot L\ ,\ ;\ ] \\
& [\ D \to \cdot,\ ;\ ] \\
& [\ L \to \cdot L\&B\ ,\ ;/\&\ ] \\
& [\ L \to \cdot B\ ,\ ;/\&\ ] \\
& [\ B \to \cdot\mathbf{id} > \mathbf{num}\ ,\ ;/\&\ ]\ \}
\end{aligned}
$$

$I_6 = \textsc{Goto}(I_5, D) = \{\ [\ S \to \mathbf{num}\ {:}D \cdot ;\ ,\ \$/\mathbf{num}\ ]\ \}$

$$
\begin{aligned}
I_7 = \textsc{Goto}(I_5, L) = \{\ & [\ D \to L\cdot\ ,\ ;\ ] \\
& [\ L \to L \cdot \&B\ ,\ ;/\&\ ]\ \}
\end{aligned}
$$

$I_8 = \textsc{Goto}(I_5, B) = \{\ [\ L \to B\cdot\ ,\ ;/\&\ ]\ \}$

$I_9 = \textsc{Goto}(I_5, \mathbf{id}) = \{\ [\ B \to \mathbf{id} \cdot > \mathbf{num}\ ,\ ;/\&\ ]\ \}$

$I_{10} = \textsc{Goto}(I_6, ;) = \{\ [\ S \to \mathbf{num}\ {:}D;\cdot\ ,\ \$/\mathbf{num}\ ]\ \}$

$$
\begin{aligned}
I_{11} = \textsc{Goto}(I_7, \&) = \{\ & [\ L \to L\& \cdot B\ ,\ ;/\&\ ] \\
& [\ B \to \cdot\mathbf{id} > \mathbf{num}\ ,\ ;/\&\ ]\ \}
\end{aligned}
$$

$I_{12} = \textsc{Goto}(I_9, >) = \{\ [\ B \to \mathbf{id}\!> \cdot\ \mathbf{num}\ ,\ ;/\&\ ]\ \}$

$I_{13} = \textsc{Goto}(I_{11}, B) = \{\ [\ L \to L\&B\cdot\ ,\ ;/\&\ ]\ \}$

$\textsc{Goto}(I_{11}, \mathbf{id}) = I_9$

$I_{14} = \textsc{Goto}(I_{12}, \mathbf{num}) = \{\ [\ B \to \mathbf{id}\!>\!\mathbf{num}\cdot\ ,\ ;/\&\ ]\ \}$

La tabla de análisis es la siguiente:

Estado	Acción							Ir-a				
	&	:	;	>	**id**	**num**	$	A	B	D	L	S
0						d3		1				2
1						d3	acc					4
2						r2	r2					
3		d5										
4						r1	r1					
5			r5		d9				8	6	7	
6			d10									
7	d11		r4									
8	r7		r7									
9				d12								
10						r3	r3					
11					d9				13			
12						d14						
13	r6		r6									
14	r8		r8									

La gramática es LR-canónica, ya que la tabla no contiene ningún conflicto.

Apartado 3. La simulación de la cadena indicada es la siguiente:

Pila	Entrada	Acción
0	**num** : **id** > ; $	d3
0 **num** 3	: **id** > ; $	d5
0 **num** 3 : 5	**id** > ; $	d9
0 **num** 3 : 5 **id** 9	> ; $	d12
0 **num** 3 : 5 **id** 9 > 12	; $	ERROR: falta **num**. Modo pánico.
		Desapilar símbolos y estados hasta 5 (transiciones en IR_A)
		Apilar B 8 (podría escogerse también L o D)
0 **num** 3 : 5 B 8	; $	$r\ L \to B$
0 **num** 3 : 5 L 7	; $	$r\ D \to L$
0 **num** 3 : 5 D 6	; $	d10
0 **num** 3 : 5 D 6 ; 10	$	$r\ S \to \mathbf{num} : D;$
0 S 2	$	$r\ A \to S$
0 A 1	$	Accept

Apartado 4. Para comprobar si la gramática es LALR debemos unificar los estados de la colección LR(1) cuyos conjuntos de ítems sean iguales, a excepción de los símbolos de anticipación. Podemos observar que no es posible unir varios estados en uno sólo, ya que todos los conjuntos de ítems son distintos. Por ello, la tabla LALR es exactamente igual a la LR-canónica, de modo que la gramática también es LALR.

Para comprobar si la gramática también es SLR debemos verificar que los conjuntos de símbolos de anticipación de los ítems que representan acciones de reducción coinciden con los conjuntos SIGUIENTE de los no terminales a la izquierda de dichos ítems.
Se calculan los conjuntos PRIMERO y SIGUIENTE:

PRIMERO(A) = $\{num\}$
PRIMERO(S) = $\{num\}$
PRIMERO(D) = $\{\lambda\ id\}$
PRIMERO(L) = $\{id\}$
PRIMERO(B) = $\{id\}$

SIGUIENTE(A) = $\{\$\ num\}$
SIGUIENTE(S) = $\{\$\ num\}$
SIGUIENTE(D) = $\{;\}$
SIGUIENTE(L) = $\{;\ \&\}$
SIGUIENTE(B) = $\{;\ \&\}$

- Para el no terminal A tenemos los ítems:
  $[\ A \to S\cdot\ ,\ \$/\mathbf{num}\ ]$ en I_1
  $[\ A \to AS\cdot\ ,\ \$/\mathbf{num}\ ]$ en I_4
  y SIGUIENTE(A) = $\{\ \$\ \mathbf{num}\ \}$.
- Para el no terminal S tenemos el ítem:
  $[\ S \to \mathbf{num} : D;\cdot\ ,\ \$/\mathbf{num}\ ]$ en I_{10}
  y SIGUIENTE(S) = $\{\ \$\ \mathbf{num}\ \}$.
- Para el no terminal D tenemos los ítems:
  $[\ D \to \cdot\ ,\ ;\ ]$ en I_5
  $[\ D \to L\cdot\ ,\ ;\ ]$ en I_7
  y SIGUIENTE(D) = $\{\ ;\ \}$.
- Para el no terminal L tenemos los ítems:
  $[\ L \to L, B\cdot\ ,\ ;/\&\ ]$ en I_{13}
  $[\ L \to B\cdot\ ,\ ;/\&\ ]$ en I_8
  y SIGUIENTE(L) = $\{\ ;\ \&\ \}$.
- Para el no terminal B tenemos el ítem:
  $[\ B \to \mathbf{id} > \mathbf{num}\cdot\ ,\ ;/\&\ ]$
  y SIGUIENTE(B) = $\{\ ;\ \&\ \}$.

Por tanto, coinciden los conjuntos SIGUIENTE con los símbolos de anticipación, de modo que la tabla SLR es igual que la LALR, sin ningún conflicto, y la gramática también es SLR.

Apartado 5. Emplearemos las siguientes funciones:

- `Lista crearLista()`: genera una lista de enteros vacía.
- `Lista crearLista(int num)`: genera una lista de enteros con un único elemento entero `num`.
- `Lista añadirElemento(Lista l, int num)`: añade el entero `num` a la lista `l`.
- `Lista unirListas(Lista l1, Lista l2)`: une las listas `l1` y `l2` en una única lista.
- `bool existeElemento(Lista l, int num)`: comprueba si el entero `num` forma parte de la lista `l`.

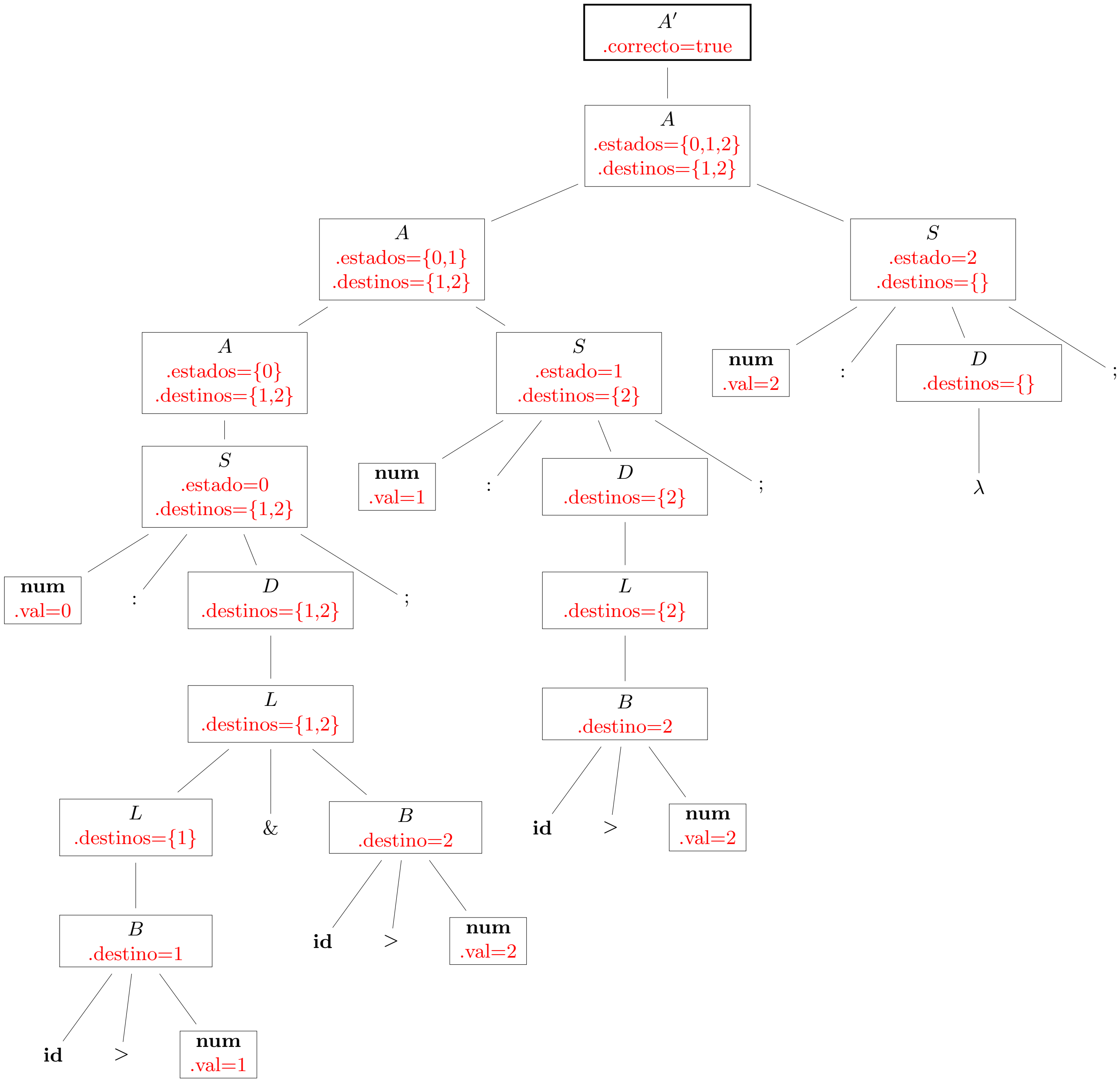
Se considera que el tipo de dato `Lista` permite iterar en sus elementos con una instrucción `foreach`. Se definen los siguientes atributos para los símbolos de la gramática:

Símbolo	**Atributo**	**Tipo**	**Comentario**
A'	correcto	`bool`	`true` si el autómata tiene los destinos definidos.
A	estados	`Lista`	Lista con los identificadores de estados de A.
	destinos	`Lista`	Lista con los destinos de todas las transiciones de A.
S	estado	`int`	Número del estado definido por S.
	destinos	`Lista`	Lista con los destinos de las transiciones que salen del estado definido por S.
D	destinos	`Lista`	Lista con los destinos de las transiciones definidas por D.
L	destinos	`Lista`	Lista con los destinos de las transiciones definidas por L.
B	destino	`int`	Número del estado destino de la transición definida por B.
num	val	`int`	Valor del lexema asociado al token **num**, en formato entero.

La definición dirigida por la sintaxis (DDS) para la comprobación de que todas las transiciones tienen estados destino correctamente definidos es la siguiente:

Regla de producción	**Acción**
$A' \rightarrow A$	A'`.correcto = true;` `foreach` e `in` A`.destinos` `if (!existeElemento(`A`.estados,`e`))` $\{A'$`.correcto = false; break;`$\}$
$A \rightarrow A_1 S$	A`.estados = unirListas(`A_1`.estados, crearLista(`S`.estado));` A`.destinos = unirListas(`A_1`.destinos, `S`.destinos);`
$A \rightarrow S$	A`.estados = crearLista(`S`.estado);` A`.destinos = `S`.destinos;`
$S \rightarrow$ **num** : D	S`.estado = `**num**`.val;` S`.destinos = `D`.destinos;`
$D \rightarrow L$	D`.destinos = `L`.destinos;`
$D \rightarrow \lambda$	D`.destinos = crearLista();`
$L \rightarrow L_1$ & B	L`.destinos = añadirElemento(`L_1`.destinos,`B`.destino);`
$L \rightarrow B$	L`.destinos = crearLista(`B`.destino);`
$B \rightarrow$ **id** $>$ **num**	B`.destino = `**num**`.val;`

El autómata del enunciado genera el siguiente árbol de análisis decorado:



Todos los atributos son **sintetizados**, ya que se obtienen a partir de los valores de los atributos de los nodos hijo. Por tanto, la gramática es **S-atribuida** y, en consecuencia, también es **L-atribuida**.