EXAMEN DE COMPILADORES (2° Grado en Informática, final julio-2015)

Apellidos, nombre: **Grupo:** **DNI:**

Instrucciones: Este enunciado y todos los folios usados deben entregarse al salir

Parte I: PREGUNTAS TIPO TEST. 30 %. Cada dos respuestas incorrectas anulan una correcta.

1. En relación con los lenguajes tratados por los compiladores, indica la respuesta **incorrecta**:
   a) La sintaxis de un lenguaje puede tener una parte libre de contexto y otra sensible al contexto.
   b) La sintaxis de todos los lenguajes de programación sólo contiene características que son libres de contexto.
   c) La verificación que se realiza al comprobar que una variable ha sido declarada previamente es un ejemplo de la sintaxis sensible al contexto de un lenguaje.

2. Elige la opción **incorrecta**. La fase de comprobación de tipos de un compilador
   a) Podría modificar el código objeto generado.
   b) Nunca podría modificar el código objeto generado.
   c) Puede realizarse en tiempo de ejecución.

3. Si un traductor genera código C a partir del código fuente, y luego se usa un compilador de C para traducir a código máquina, podemos considerar que tenemos:
   a) Un compilador cruzado.
   b) Un compilador de varias pasadas.
   c) Un compilador de varias etapas.

4. Dado el siguiente fragmento de un fichero *flex*:

```
letra [a-zA-Z]
digito[0-9]
%%
[ \n\t] ;
"=" return EQ;
"+" return MAS;
"-" return MENOS;
"*" return POR;
"/" return DIV;
";"  return PUNTOYCOMA;
({letra}|"_")({letra}|{digito}|"_")*   { return ID;}
[^a-zA-Z0-9;*+=/-]*     print ("Error carácter no reconocido %s", yytext);
```

   indica cual sería la primera salida por pantalla del analizador léxico teniendo en cuenta la siguiente cadena

```
?__id10 = var_1 + c;
```

   a) `Error carácter no reconocido ?_`
   b) `Error carácter no reconocido ?__`
   c) `Error carácter no reconocido ?`

5. Dada la siguiente gramática

$$\begin{array}{lcl} S & \rightarrow & S\ A \mid A \\ A & \rightarrow & id\ =\ L\ ; \\ L & \rightarrow & id \mid L\ =\ L \end{array}$$

   y la forma sentencial *a=b=c=d=e;*
   a) Existe sólo un árbol de derivación para la anterior forma sentencial.
   b) Existen sólo dos árboles de derivación para la anterior forma sentencial.
   c) Existen cuatro árboles de derivación para la anterior forma sentencial.

6. Elige de entre las siguientes, la frase que consideres **incorrecta**:

   a) Una gramática *no recursiva por la izquierda* podría ser también *no propia*.

   b) Una gramática *ambigua* puede ser LR aunque no LL.

   c) Una gramática que sea a la vez LL y LR puede ser *no propia*.

7. Si factorizamos siguiente gramática

$$\begin{array}{lcl} declarations & \rightarrow & declarations\ var\ identifierL \\ & & \mid declarations\ let\ identifierL \\ & & \mid \lambda \\ identifierL & \rightarrow & asig \\ & & \mid identifierL\ ,\ asig \\ asig & \rightarrow & id \end{array}$$

   obtenemos:

   a)

$$\begin{array}{lcl} declarations & \rightarrow & declarations\ declarations' \mid \lambda \\ declarations' & \rightarrow & var\ identifierL \mid let\ identifierL \\ identifierL & \rightarrow & asig \mid identifierL\ ,\ asig \\ asig & \rightarrow & id \end{array}$$

   b)

$$\begin{array}{lcl} declarations & \rightarrow & declarations\ declarations' \mid \lambda \\ declarations' & \rightarrow & declarations''\ identifierL \\ declarations'' & \rightarrow & var \mid let \mid \lambda \\ identifierL & \rightarrow & asig \mid identifierL\ ,\ asig \\ asig & \rightarrow & id \end{array}$$

   c)

$$\begin{array}{lcl} declarations & \rightarrow & declarations\ declarations' \\ declarations' & \rightarrow & var\ identifierL \mid let\ identifierL \mid \lambda \\ identifierL & \rightarrow & asig \mid identifierL\ ,\ asig \\ asig & \rightarrow & id \end{array}$$

8. Dada la siguiente gramática:

$$\begin{array}{lcl} P & \rightarrow & D\ L \\ D & \rightarrow & \lambda \mid D\ T\ id\ ; \\ T & \rightarrow & int \mid float \\ L & \rightarrow & L\ S \mid \lambda \\ S & \rightarrow & print\ id\ ; \mid id\ =\ num\ ; \end{array}$$

   ¿cuál de los siguientes conjuntos es incorrecto?

   a) SIGUIENTE(D) = $\{int, float, print, id\}$

   b) PRIMERO(L) = $\{\lambda, print, id\}$

   c) SIGUIENTE(L) = $\{print, id, \$\}$

9. Dada la gramática

$$E \rightarrow E\ +\ E \mid E\ *\ E \mid num$$

   y la forma sentencial `E + E * num`

   a) El pivote es `E + E`.

   b) El pivote es `E * num`.

   c) Tiene dos pivotes.

10. Dada la gramática G siguiente:

$$\begin{array}{lll} S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ E \mid \texttt{print str} \\ C & \rightarrow & \texttt{id == num} \\ E & \rightarrow & \texttt{else}\ S \end{array}$$

Para reconocer la sentencia

`if x == 7 then print ‘‘Sí es 7’’ else print ‘‘No es 7’’`

en un análisis ascendente predictivo, la primera regla con la que se reduciría sería:

a) $S \rightarrow$ `if` C `then` $S\ E$

b) $S \rightarrow$ `print str`

c) $C \rightarrow$ `id == num`

11. Dada la siguiente gramática

$$\begin{array}{lll} S & \rightarrow & S\ A \mid A \\ A & \rightarrow & id\ =\ L\ ; \\ L & \rightarrow & id \mid L\ =\ L \end{array}$$

podemos afirmar

a) la gramática es LL.

b) la grámatica ni es LL ni es LR.

c) la gramática es SLR, pero no LL puesto que es recursiva por la izquierda.

12. Dada la siguiente gramática G,

$$\begin{array}{lll} S & \rightarrow & \texttt{if num}\ S\ E \mid \texttt{print str} \\ E & \rightarrow & \texttt{else}\ S \end{array}$$

ante la entrada `if num str else print str`, el método de análisis descendente predictivo comenzaría realizando los siguientes pasos:

Pila	Entrada	Salida
$ S	if num str else print str $	
$ E S num if	if num str else print str $	S → if num S E
$ E S num	num str else print str $	
$ E S	str else print str $	

Indicar cuál sería la configuración siguiente, suponiendo una recuperación de errores en modo pánico:

a)

Pila	Entrada	Salida
$ E S	print str else print str $	

b)

Pila	Entrada	Salida
$ E	str else print str $	

c)

Pila	Entrada	Salida
$ E	else print str $	

13. Dada la siguiente gramática:

$$\begin{array}{lll} S & \rightarrow & X\ y\ Y \\ X & \rightarrow & x\ \alpha \mid \lambda \\ Y & \rightarrow & X\ x\ \beta \end{array}$$

donde α y β son cadenas de terminales y no terminales:

a) No puede ser LL(1) ni SLR(1).

b) Puede ser LL(1) pero no SLR(1).

c) Puede ser SLR(1) pero no LL(1).

14. Supongamos que se realiza un análisis SLR de la gramática siguiente

$$L \ \rightarrow L \vee L \mid \neg L \mid L \Rightarrow L \mid (L) \mid \texttt{true} \mid \texttt{false}$$

y uno de los conjuntos de items es el siguiente:

$I_{11} = \{ [\ L \rightarrow L\ \vee L \bullet\ ]\ ,\ [\ L \rightarrow L\ \bullet\ \vee\ L\ ]\ ,\ [\ L\ \rightarrow L \bullet\ \Rightarrow\ L\ ]\ \}$

de manera que, en la tabla de análisis la fila correspondiente al estado 11 quedaría así:

Estado	Acción						Ir-a
	$\vee$	$\neg$	$\Rightarrow$	`true`	`false`	\$	L
11	r1/d5		r1/d7			r1	

Teniendo en cuenta que todos los operadores son asociativos por la izquierda y que $\Rightarrow$ tiene menor precedencia que $\vee$, para eliminar los conflictos deberíamos:

a) Elegir la reducción en la casilla $[11, \vee]$ y el desplazamiento en la casilla $[11, \Rightarrow]$.

b) Elegir la reducción en ambas casillas .

c) Elegir el desplazamiento en ambas casillas.

15. Dada la siguiente DDS

$$\begin{array}{lcl} declarations & \rightarrow & declarations_1\ var\ identifierL\ \{identifierL.h = var\} \\ & & \mid declarations_1\ let\ identifierL\ \{identifierL.h = let\} \\ & & \mid \lambda \\ identifierL & \rightarrow & asig\ \{asig.h = identifierL.h\} \\ & & \mid identifierL_1\ ,\ asig\ \{identifierL_1.h = identifierL.h; asig.h = identifierL.h\} \\ asig & \rightarrow & id\ \{instertaIdTipo(id.val, asig.h)\} \end{array}$$

indica la respuesta correcta:

a) h es un atributo heredado en los símbolos no terminales identifierL y asig que indica los valores *var* para una variable y *let* para una constante.

b) h es un atributo sintetizado en los símbolos no terminales identifierL y asig que indica los valores *var* para una variable y *let* para una constante.

c) h es un atributo heredado en el símbolo no terminal identifierL y h es un atributo sintetizado en asig, indicando los valores *var* para una variable y *let* para una constante.

Parte II: PROBLEMA. 70 %.

La siguiente gramática G con $V_T = \{\vee, \forall, \texttt{id}, \texttt{(}, \texttt{)}, \texttt{,}\}$ y $V_N = \{F, L\}$, siendo P:

$$\begin{array}{ll} F & \rightarrow F\ \vee\ F \mid \forall\ \texttt{id (} F \texttt{)} \mid \texttt{id (L)} \\ L & \rightarrow \texttt{id} \mid \texttt{id ,}\ L \end{array}$$

donde el operador $\vee$ es asociativo por la izquierda, permite representar parcialmente fórmulas de lógica de predicados. Se pide:

1. *(1 punto)* Calcular los conjuntos PRIMERO y SIGUIENTE de F y L. Sin calcular los conjuntos *predict* ni la tabla LL, dar todos los argumentos para justificar que G no es una gramática LL(1).

2. *(1.75 puntos)* Modificar la gramática G para intentar conseguir una equivalente que sea LL(1). Comprobar si la nueva gramática es LL(1) calculando los conjuntos *predict* para cada regla.

3. *(2 puntos)* Indicar y justificar si la gramática G es SLR(1), LR(1) y/o LALR(1), calculando la colección LR(0) y la tabla de análisis SLR. En caso de que no sea SLR, eliminar en la tabla los conflictos de forma adecuada.

4. *(0.5 puntos)* Simular, con la tabla obtenida al eliminar los conflictos, el algoritmo ascendente con la entrada `∀id(∨id(id))` haciendo recuperación en modo pánico en caso de error.

5. *(1.75 puntos)* Suponiendo que se puede consultar el lexema asociado al token `id`:

   a) Realizar una definición dirigida por la sintaxis (DDS) que permita obtener la lista de variables no ligadas por cuantificadores $\forall$ en una fórmula cualquiera que se pueda generar con G. Por ejemplo, en la fórmula `∀x(f(x,y)∨∀z(g(z,u,x)))`, la lista de variables no ligadas es $\{\texttt{y}, \texttt{u}\}$.

   b) Decorar el árbol de análisis para la entrada `∀x(f(x,y)∨g(u,x))`.

   c) ¿La gramática G es L-Atribuida? ¿Y S-Atribuida? Justifica las respuestas.