EXAMEN DE COMPILADORES (2° Grado en Informática, final junio-2016)

Apellidos, nombre: **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. ¿Cuál de los siguientes tipos de reglas de producción corresponden a las gramáticas que describen a la mayoría de los lenguajes de programación?

   a) $A \to a$, $A \to aB$ o $A \to \lambda$, con $A \in V_N$ y $a \in V_T$.

   b) $A \to \alpha$, con $A \in V_N$ y $\alpha \in (V_N \cup V_T)^*$.

   c) $\alpha A \beta \to \alpha\gamma\beta$, con $A \in V_N$, $\alpha\beta \in (V_N \cup V_T)^*$ y $\gamma(V_N \cup V_T)^+$.

2. Indica cual de las siguientes afirmaciones es cierta

   a) El análisis léxico analiza todo el código fuente y devuelve todos los tokens al analizador sintáctico para que éste pueda comenzar el análisis.

   b) La tabla de símbolos es una estructura que guarda todos los tokens devueltos por el analizador léxico.

   c) El analizador sintáctico invoca al analizador léxico sólo cuando necesita un token del código fuente, puesto que es el analizador léxico el que lee los caracteres del código fuente.

3. Con respecto al ánalisis semántico, indica cual NO es una de sus funcionalidades

   a) Comprobación del uso correcto de las variables.

   b) Obtención del código objeto.

   c) Evaluación de los atributos de los simbolos de la gramática.

4. Indica cual de las siguientes afirmaciones es cierta

   a) Una máquina virtual de Linux alojada en un sistema Windows ejecutará las instrucciones más lentamente que un PC con el sistema operativo Linux.

   b) Un programa ensamblador tendrá una portabilidad bastante alta.

   c) Un intérprete resulta muy útil cuando las instrucciones se van a ejecutar muchas veces.

5. Supongamos una especificación de `bison` con la siguiente declaración de tipos:

```
%union {
        char * cadena;
        double numero;
}
%token <cadena> ID
%token <numero> FLT
```

   ¿qué instrucciones serían correctas en la acción asociada a FLT en un analizador léxico implementado con Flex?

   a) {`yylval.numero = atoi(yytext); return FLT;` }

   b) {`yylval.cadena = strdup(yytext); return FLT;` }

   c) {`yylval.numero = atof(yytext); return FLT;` }

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

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

   indica cual de las siguientes afirmaciones es correcta en el reconocimiento de la forma sentencial $\forall$`id(`$\forall$`id(id))`:

   a) Con una gestión de error en modo pánico se detectará un error léxico, puesto que el token `id` no puede ser reconocido entre paréntesis.

   b) Esta forma sentencial no pertenece al lenguaje generado por la gramática porque presenta errores sintácticos y la gestión de errores en modo pánico sólo podría avisar del error y permitir continuar con el análisis.

   c) No existe ningún error en el reconocimiento de esa forma sentencial pues pertenece al lenguaje generado por la gramática.

7. Dada la siguiente gramática:

   $A \to B\ C\ A \mid a$
   $B \to b \mid \lambda$
   $C \to A\ B\ c \mid c$

   a) Tiene ciclos.

   b) No está factorizada.

   c) Es recursiva por la izquierda.

8. Dada la siguiente gramática

$$\begin{array}{lcl} T & \to & [\ T\ ] \mid T\ \mapsto\ T \mid (\ listaT\ ) \mid Integer \mid Float \\ listaT & \to & T,\ listaT \mid \lambda \end{array}$$

   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.

9. Dada la siguiente gramática

$$\begin{array}{lcl} T & \to & [\ T\ ] \mid T\ \mapsto\ T \mid (\ listaT\ ) \mid Integer \mid Float \\ listaT & \to & T,\ listaT \mid \lambda \end{array}$$

   el conjunto I_0 de la colección LR(1) es:

   a) $\{[T' \to \cdot T, \$], [T \to \cdot[T], \$], [T \to \cdot T\ \mapsto\ T, \$/\ \mapsto], [T \to \cdot(listaT), \$/\ \mapsto], [T \to \cdot Integer, \$],$
   $[T \to \cdot Float, \$]\}$

   b) $\{[T' \to \cdot T, \$], [T \to \cdot[T], \$/\ \mapsto], [T \to \cdot T\ \mapsto\ T, \$/\ \mapsto], [T \to \cdot(listaT), \$/\ \mapsto], [T \to \cdot Integer, \$/\ \mapsto],$
   $[T \to \cdot Float, \$/\ \mapsto]\}$

   c) $\{[T' \to \cdot T, \$], [T \to \cdot[T], \$/\ \mapsto], [T \to \cdot T\ \mapsto\ T, \$/\ \mapsto], [T \to \cdot(listaT), \$/\ \mapsto]\}$

10. Dada la siguiente gramática:

$$\begin{array}{ll} X & \to Y\ Z \\ Y & \to a\ Y \mid \lambda \\ Z & \to T \mid \lambda \\ T & \to T\ a \mid a \end{array}$$

    a) El conjunto $PRIMERO$ para todos los no terminales es el mismo, pues todas las sentencias empiezan por a.

    b) El conjunto $SIGUIENTE$ de Y es $\{a, \lambda\}$.

    c) El conjunto $SIGUIENTE$ de T es $\{a, \$\}$.

11. Dada la siguiente gramática:

$$\begin{array}{ll} X & \to Y\ Z \\ Y & \to a\ Y \mid \lambda \\ Z & \to T \mid \lambda \\ T & \to T\ a \mid a \end{array}$$

    el pivote de la forma sentencial derecha $w \equiv Y$ es:

    a) a.

    b) Z.

    c) λ.

12. Si una misma forma sentencial derecha tiene más de un pivote, entonces:

    a) La gramática es recursiva por la derecha.

    b) La gramática es ambigua.

    c) La forma sentencial derecha es incorrecta desde el punto de vista sintáctico.

13. Si un compilador de C tiene que procesar una par de líneas de código como las siguientes:

```
int a = 5;
float c = 2.0 * a;
```

   a) Realiza una *coerción* de `int` a `float` sin informar al programador.

   b) Indica un error semántico para obligar al programador a hacer un *casting*.

   c) No tiene que hacer ninguna conversión para llevar a cabo la generación de código.

14. ¿Qué tipo de representación intermedia es más conveniente para la optimización de expresiones aritméticas con subexpresiones comunes, como $a + a * (b - c) + (b - c) * d$?

   a) Grafos dirigidos acíclicos.

   b) Árboles sintácticos.

   c) Cuádruplas.

15. Dada la siguiente definición dirigida por la sintaxis:

$$\begin{array}{lll} P & \rightarrow [L](num) & \{L.x = num.v; P.v = L.v;\} \\ L & \rightarrow E & \{L.v = E.v;\} \\ L & \rightarrow L_1, E & \{L_1.x = L.x; L.v = L_1.v * L_1.x + E.v;\} \\ E & \rightarrow num & \{E.v = num.v;\} \\ E & \rightarrow P & \{E.v = P.v;\} \end{array}$$

   podemos asegurar que esta gramática atribuida:

   a) No es ni L-Atribuida ni S-atribuida.

   b) Es L-Atribuida, aunque no S-Atribuida pues tiene atributos heredados.

   c) Es L-Atribuida y, por tanto, S-Atribuida, puesto que los atributos cumplen los requisitios de ambas.

Parte II: PROBLEMA. 70 %.

Considerar la siguiente gramática G con $V_T = \{\cup, \cap, \{\ , \}\ , (,), \textbf{,}, el, \emptyset\ \}$ y $V_N = \{E, C, L\}$, siendo P:

$$\begin{array}{lcl} E & \rightarrow & C \\ & | & C \cup E \\ & | & C \cap E \\ C & \rightarrow & \{\ L\ \} \\ & | & (\ E\ ) \\ & | & \emptyset \\ L & \rightarrow & el \\ & | & el\ ,\ L \end{array}$$

que genera un lenguaje para operar con conjuntos. Por ejemplo, la gramática podría generar la siguiente cadena:

$$\{1, 7\} \cup (\{1, 2, 3, 4, 5, 6\} \cap \{3, 5\})$$

que representaría al conjunto $\{1, 3, 5, 7\}$.

1. *(2.5 puntos)* Decir si se trata de una gramática LL(1), justificando la respuesta. Si no se tratara de una gramática LL(1), intentar transformarla para conseguir que lo sea. Calcular los conjuntos PRIMERO y SIGUIENTE para cada no terminal y *predict* para cada regla. Construir la tabla de análisis y razonar si la gramática es LL(1).

2. *(1.25 puntos)* Sobre la gramática G original calcular la colección LR(0) y la tabla de análisis SLR.

3. *(0.5 puntos)* ¿Es la gramática SLR? ¿Y LALR? ¿Y LR-Canónica?

4. *(1 punto)* Simular el comportamiento de algoritmo ascendente predictivo para reconocer la cadena $\{a, b\}\}$, aplicando el método de *recuperación de errores en modo pánico* en caso de error.

5. *(1.75 puntos)* Supongamos que trabajamos con conjuntos de elementos de tipo `cadena`, `id` o `número`. Es decir, el token *el* tiene un atributo denominado *tipo* que puede tomar uno de esos tres valores. No se permite la pertenencia de elementos de diferentes tipos en un mismo conjunto. Realizar una *definición dirigida por la sintaxis* para comprobar esto. También debe comprobarse que no se opera (con $\cup$ o $\cap$) con conjuntos de tipos diferentes. En cualquiera de los dos casos debe producirse un error de tipos. Para ello definir los atributos que sean necesarios, indicando si son sintetizados o heredados. Decorar el árbol sintáctico correspondiente a la entrada

$$\{\text{“}cad1\text{”}, \text{“}cad2\text{”}\} \cup (\{a, b\} \cap \{b\})$$

   Decir si la gramática es S-Atribuida y/o L-Atribuida.