EXAMEN DE COMPILADORES (2° Grado en Informática, final julio-2014)

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. La simulación de un autómata de pila:
   a) Permite el reconocimiento de todos los aspectos de cualquier lenguaje de programación.
   b) No permite el reconocimiento de algunos aspectos de la mayoría de los lenguajes de programación.
   c) No se usa en la fase de análisis de los compiladores.

2. En las prácticas de la asignatura de este curso, para poder ejecutar el código MIPS generado:
   a) Usamos un programa ensamblador, que traduce desde el lenguaje ensamblador a lenguaje máquina.
   b) Usamos una máquina virtual, que interpreta el lenguaje ensamblador generado.
   c) El código MIPS generado se ejecuta directamente en nuestra máquina.

3. Un intérprete y un traductor:
   a) Requieren las mismas entradas y generan las mismas salidas.
   b) El intérprete necesita, además del programa fuente, la entrada para ese programa.
   c) El traductor genera los resultados de la ejecución del programa fuente.

4. Supongamos que en un lenguaje los identificadores pueden estar formados por letras minúsculas, dígitos y guiones bajos, pero no pueden empezar ni terminar por guión bajo, ni llevar dos guiones seguidos, y, obligatoriamente han de contener alguna letra. Por ejemplo, son correctos: "abc", "123a", "a_b"; son incorrectos: "_a", "a_", "a___b", "1_2", "369". Elegir una expresión regular adecuada en Flex para reconocerlos.
   a) $([a-z0-9]_ \mid [a-z0-9])* \ [a-z] \ (_[a-z0-9] \mid [a-z0-9])*$
   b) $([\hat{}_]_ \mid [a-z0-9])* \ [a-z] \ (_[\hat{}_] \mid [a-z0-9])*$
   c) $[a-z0-9]+ \ (_[a-z0-9] \mid [a-z0-9])* \ [a-z0-9]+$

5. Supongamos que la tabla de análisis LL para una gramática dada es la siguiente:

	a	\$
A	$A \to a\ A\ a$ $A \to \lambda$	$A \to \lambda$

   a) La gramática no es LL(1) porque no es λ-libre.
   b) La gramática no es LL(1) porque $predict(A \to a\ A\ a) \cap predict(A \to \lambda) = \{a\}$.
   c) La gramática sí es LL(1), sólo que en la tabla falta la columna para el símbolo λ.

6. Dada la siguiente gramática:

$$\begin{array}{lcl} X & \to & Y\ Z\ X \mid x \mid \lambda \\ Y & \to & X\ Y\ Z \mid y \mid \lambda \\ Z & \to & Z\ Y\ X \mid z \mid \lambda \end{array}$$

   Elige la opción **INCORRECTA**:
   a) PRIMERO(X)=PRIMERO(Y)=PRIMERO(Z).
   b) SIGUIENTE(X)=SIGUIENTE(Y)=SIGUIENTE(Z).
   c) SIGUIENTE(Y)={x,y,z}.

7. La gramática anterior:
   a) Es LL(1) y SLR(1).
   b) Es SLR(1) pero no LL(1).
   c) No es LL(1) ni SLR(1).

8. La siguiente gramática:

$$\begin{array}{rcl} S & \rightarrow & a\ S\ a \\ & | & \lambda \end{array}$$

   a) Es ambigua, no LL y no LR(1).

   b) Es no ambigua, no LL y no LR(1).

   c) Es no ambigua, no LL y LR(1).

9. Dada la siguiente gramática:

$$\begin{array}{rcl} S & \rightarrow & sent \\ & | & S; S \\ & | & \lambda \end{array}$$

   ante una frase como `sent ; sent ; sent`, un analizador LR, al leer los tres primeros tokens se encontrará con el dilema de reducir con la segunda regla o desplazar el token `;`. Si tenemos en cuenta que se usa el `;` como separador de sentencias y queremos optimizar el uso de la memoria de la pila:

   a) Sería más correcto desplazar.

   b) Sería más correcto reducir.

   c) Sería indiferente.

10. Supongamos que una gramática LALR tiene entre sus conjuntos de ítems el siguiente:

$$\begin{array}{l} I_i = \{\ [\ A \rightarrow \alpha\ \cdot\ a\ C\ ,\ c/d\ ] \\ \qquad\quad [\ C \rightarrow \gamma\ \cdot\ ,\ c/d\ ]\ \} \end{array}$$

    Indica la opción correcta:

    a) Si SIGUIENTE$(C) = \{c, d\}$, la gramática no puede ser SLR.

    b) Si SIGUIENTE$(C) = \{c, d, a\}$, la gramática no puede ser SLR.

    c) Si SIGUIENTE$(C) = \{c, d, b\}$, la gramática no puede ser SLR.

11. La gramática:

$$\begin{array}{rcl} S & \rightarrow & id := E \\ & | & if\ E = false\ then\ S \\ E & \rightarrow & id \\ & | & E \leq E \leq E \end{array}$$

    a) Es propia, no recursiva por la izquierda y LL.

    b) Es propia, no LL y LR.

    c) No es LR.

12. Supongamos que, en el instante actual, la configuración de un analizador ascendente es la siguiente:

Pila	Entrada
$0\ T\ 2\ *\ 7\ F\ 10$	$+\ id\ \$$

    ¿Cuál de las siguientes es la forma sentencial derecha actual en el proceso de análisis sintáctico?

    a) $T\ *\ F\ +\ id$

    b) $0\ T\ \ 2\ *\ 7\ F\ 10\ +\ id$

    c) $T\ *\ F$

13. Dada la gramática:

$$\begin{array}{rcl} S & \rightarrow & id := E \\ & | & if\ E = false\ then\ S \\ E & \rightarrow & id \\ & | & E \leq E \leq E \end{array}$$

    En la aceptación de la cadena `if id` $\leq$ `id` $\leq$ `id = false then id := id` , con un analizador LR serían necesarios:

    a) 12 desplazamientos y 7 reducciones.

    b) 12 desplazamientos y 5 reducciones.

    c) 13 desplazamientos y 13 reducciones.

14. Dado un item $[A \to \alpha\cdot, a/b/c]$ perteneciente a una colección **LALR** para cierta gramática, donde α sería una cadena formada por símbolos tanto terminales como no terminales:

    a) El conjunto de tokens de anticipación $\{a, b, c\}$ será siempre un subconjunto propio de los que aparezcan en cualquier item perteneciente a la colección **LR(1)** cuya parte izquierda sea idéntica.

    b) El conjunto de tokens de anticipación $\{a, b, c\}$ contendrá siempre a los tokens que aparezcan en cualquier item perteneciente a la colección **LR(1)** cuya parte izquierda sea idéntica.

    c) El conjunto de tokens pertenecientes a $SIGUIENTE(A)$ será siempre un subconjunto propio de los tokens de anticipación $\{a, b, c\}$.

15. Dada la siguiente *definición dirigida por la sintaxis*:

$$\begin{array}{lcll} S & \to & L & S.t = L.t \\ L & \to & L_1, S & S.t = f(L_1.t) \\ & & & L.t = L_1.t||S.t \\ L & \to & S & L.t = S.t \\ S & \to & a & S.t = g(a.val) \end{array}$$

    a) La gramática es L-atribuida y por tanto S-atribuida.

    b) La gramática es L-atribuida pero no S-atribuida.

    c) La gramática no es L-atribuida ni S-atribuida.

Parte II: PREGUNTAS CORTAS. 10 %.
Sea la siguiente gramática con atributos:

$$\begin{array}{lcll} S & \to & A\ B\ C & B.u = S.u \\ & & & A.u = B.v + C.v \\ & & & S.v = A.v \\ A & \to & a & A.v = 2 * A.u \\ B & \to & b & B.v = B.u \\ C & \to & c & C.v = 1 \end{array}$$

Se pide:

1. Dibujar el árbol de análisis para la cadena *abc* y dibujar el *grafo de dependencia* para los atributos asociados.
2. Si añadimos la acción semántica $S.u = 3$ a la regla de producción $S \to A\ B\ C$ ¿Cuál sería el valor de $S.v$ al terminar la evaluación?

Parte III: PROBLEMA. 60 %.

Considerar la siguiente gramática G:

$$\begin{array}{lcl} S & \rightarrow & id = E \\ E & \rightarrow & id \\ & | & id(E, E, E) \end{array}$$

que genera un subconjunto de un lenguaje de programación:

1. *(1.5 puntos)* Decir si se trata de una gramática LL(1), justificando la respuesta. Si no se tratara de una gramática LL(1), intentar transformarla en otra gramática equivalente G' que sí lo sea. Calcular los conjuntos PRIMERO y SIGUIENTE para cada no terminal de G' y *predict* para cada regla de G'. Razonar con estos últimos conjuntos si G' es LL(1), sin construir la tabla de análisis.

2. *(1.75 puntos)* Calcular la colección LR(0) para G y la tabla de análisis para comprobar si se trata de una gramática SLR.

3. *(0.5 puntos)* Simular el comportamiento de algoritmo ascendente predictivo para reconocer la cadena

   ```
   id = id ( id , id id , )
   ```

   aplicando el método de recuperación de errores en modo pánico en caso de error.

4. *(1 punto)*

   a) ¿Es la gramática LALR? ¿Y LR-Canónica?

   b) ¿Tendrán las tablas SLR y LALR el mismo tamaño? Y la LALR, ¿tendrá el mismo tamaño que la LR-Canónica? Razona estas respuestas.

   c) Supongamos que aumentamos la gramática con dos reglas más, que permiten la generación de múltiples sentencias:

   $$\begin{array}{lcl} S & \rightarrow & id = E \\ & | & S; S \\ & | & \lambda \\ E & \rightarrow & id \\ & | & id(E, E, E) \end{array}$$

   ¿Sería en este caso la gramática LALR y/o LR-Canónica?

5. *(1.25 puntos)* Diseñar un *comprobador de tipos* para el lenguaje generado por G, mediante una *definición dirigida por la sintaxis*, teniendo en cuenta que:

   - El lexema del token `id` correspondiente a la última regla de producción, se puede instanciar con dos lexemas, es decir, se dispone de dos funciones: `dosT` y `potMul`.
   - Pueden usarse funciones tanto para acceder a la *tabla de símbolos* como para actualizarla.
   - Hay identificadores lógicos y naturales.
   - La expresión `dosT (`E_1`, `E_2`, `E_3`)` es verdadera solo si exactamente dos de sus argumentos se evalúan a verdadero y el otro a falso. Solo se aplica a expresiones lógicas y el resultado es de tipo lógico.
   - La expresión `potMul (`E_1`, `E_2`, `E_3`)` es equivalente a $E_1 * E_2^{E_3}$. Se aplica a expresiones naturales o lógicas y el resultado es de tipo natural.
   - No hay conversión implícita de tipos.