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

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. Un lenguaje de programación en el que se requiera la declaración de variables previa a su uso **no** puede ser:
   a) *Libre de contexto.*
   b) *Sensible al contexto.*
   c) De *estructura de frase.*
2. Una *máquina virtual*:
   a) Traduce el código intermedio a código máquina.
   b) Es un intérprete para un lenguaje de bajo nivel.
   c) Es un intérprete para un lenguaje de alto nivel.
3. Elige la respuesta **incorrecta** acerca de los *compiladores interpretados*:
   a) No generan código máquina ni ensamblador.
   b) Generan una salida que se ejecutará a más velocidad que la generada por un compilador normal.
   c) Generan una salida más portable que la generada por un compilador normal.
4. Dada una especificación léxica de Flex con la siguiente lista de expresiones regulares:
   $(aa)*$
   $ab+$
   $ba+$
   ¿Cuál de las siguientes cadenas no se podría procesar satisfactoriamente?
   a) *aaabbbab*
   b) *abbaabaa*
   c) *abbbbba*
5. En un lenguaje con las palabras clave `if`, `while`, `for`, con identificadores que comiencen siempre por letra o dígito, números sólo naturales y los operadores `+`, `-`, `*`, `/` y `=`, la expresión regular que describe en formato Flex las cadenas erróneas es:
   a) $[\hat{}a-zA-Z0-9+-=*/]+$.
   b) $[\hat{}0-9+=a-z*/A-Z-]+$.
   c) $[a-zA-Z0-9+=*/]+$.
6. Considérese la siguiente gramática, recursiva por la izquierda:
   $S \to X\ a \mid c$
   $X \to S\ b$
   ¿Cuál de las siguientes gramáticas es el resultado de eliminar correctamente la recursividad?
   a) $S \to X\ a \mid c$
      $X \to c\ b \mid X\ a\ b$
   b) $S \to a\ X \mid c$
      $X \to c\ b \mid c\ b\ X'$
      $X' \to a\ b \mid \lambda$
   c) $S \to X\ a \mid c$
      $X \to c\ b \mid c\ b\ X'$
      $X' \to a\ b \mid a\ b\ X'$

7. La siguiente gramática:

$$\begin{array}{lcl} S & \rightarrow & A\ a\ y \\ A & \rightarrow & a\ x \\ & | & \lambda \end{array}$$

   a) Es LL(1).

   b) No es SLR(1).

   c) No es LL(1) pero sí SLR(1).

8. La gramática:

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

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

   b) Es propia, no LL y no LR.

   c) Puede ser LR.

9. Una gramática **no** *propia*:

   a) Puede ser *LL* y *LR*.

   b) No puede ser *LL*, aunque sí *SLR*.

   c) No puede ser *SLR*, aunque sí *LR-Canónica*.

10. Supongamos que hemos calculado la colección $LR(1)$ para la gramática:

$$\begin{array}{lcl} S & \rightarrow & aAd \mid bBd \mid aBe \mid bAe \\ A & \rightarrow & c \\ B & \rightarrow & c \end{array}$$

    de modo que los conjuntos I_6 e I_9 contienen los siguientes items:

$$I_6 = \{[A \rightarrow c\ \bullet, d], [B \rightarrow c\ \bullet, e]\}$$

$$I_9 = \{[A \rightarrow c\ \bullet, e], [B \rightarrow c\ \bullet, d]\}$$

    Sabiendo que la gramática es *LR-canónica*, indica la respuesta correcta:

   a) La gramática es $LALR$ y SLR.

   b) La gramática es $LALR$ aunque no tiene por qué ser SLR.

   c) La gramática no es $LALR$ ni SLR.

11. Supongamos que estamos creando la colección LR(1) de una gramática, y tenemos que aplicar la operación de clausura al ítem $[\ S \rightarrow\ A \cdot X\ Y\ Z\ ,\ s\ ]$. La gramática tiene las siguientes reglas de producción:

    $X \rightarrow x_1 \mid x_2$
    $Y \rightarrow y \mid \lambda$
    $Z \rightarrow z \mid \lambda$

    ¿Cuáles de los siguientes ítems serían añadidos por aplicación de la operación clausura?

   a) $[\ X \rightarrow \cdot x_1\ ,\ y], [\ X \rightarrow \cdot x_2\ ,\ y]$

   b) $[\ X \rightarrow \cdot x_1\ ,\ x_1/x_2], [\ X \rightarrow \cdot x_2\ ,\ x_1/x_2\ ]$

   c) $[\ X \rightarrow \cdot x_1\ ,\ y/z/s\ ], [\ X \rightarrow \cdot x_2\ ,\ y/z/s\ ]$

12. Dada la siguiente gramática:

    $E \rightarrow T * E \mid T$
    $T \rightarrow n + T \mid n \mid (E)$

    ¿Cuál es el pivote de la forma sentencial derecha $((n + n) * n)$:

   a) $((\underline{n} + n) * n)$

   b) $((n + \underline{n}) * n)$

   c) $((n + n) * \underline{n})$

13. Dada la gramática de la pregunta anterior, y la cadena de entrada $((n + n) * n)$, las tres primeras reducciones que realizaría un analizador LR serían:

    a) $1^a)\ E \to T;\ 2^a)\ T \to n + T;\ 3^a)\ T \to n.$

    b) $1^a)\ T \to n + T;\ 2^a)\ T \to n;\ 3^a)\ E \to T.$

    c) $1^a)\ T \to n;\ 2^a)\ T \to n + T;\ 3^a)\ E \to T.$

14. Dado el siguiente esquema de traducción para calcular el desplazamiento en memoria de las variables:

$$
\begin{array}{lcll}
P & \to & & \{desplazamiento = 0;\} \\
 & & D & \\
D & \to & T\ \textbf{id}\ ; & \{agregarTipo(id.entrada, T.tipo, desplazamiento); \\
 & & & desplazamiento = desplazamiento + T.anchura;\} \\
 & & D_1 & \\
D & \to & \lambda & \\
T & \to & \textbf{int} & \{T.tipo = integer; T.anchura = 4;\} \\
T & \to & \textbf{float} & \{T.tipo = float; T.anchura = 8;\}
\end{array}
$$

    ¿Cuál sería el desplazamiento de la variable `x` en el siguiente bloque de declaraciones?

    ```
    int y; float z; int x;
    ```

    a) 12

    b) 8

    c) 4

15. El siguiente esquema de traducción:

$$
\begin{array}{lcl}
S & \to & A\ \{B.c = A.c\}\ B \\
A & \to & a\ A_1\ \{A.c = A_1.c + 1\} \\
A & \to & a\ \{A.c = 1\} \\
B & \to & b\ \{B_1.c = B.c - 1\}\ B_1 \\
B & \to & b\ \{if\ (B.c - 1 = 0)\ printf(True); \\
 & & \qquad\qquad else\ printf(False);\}
\end{array}
$$

    a) Sólo usa atributos sintetizados y, por tanto, es S-Atribuida.

    b) Usa atributos tanto heredados como sintetizados, aunque no es L-Atribuida, pues algunos atributos heredan del hermano derecho.

    c) Usa atributos tanto heredados como sintetizados, y es L-Atribuida.

Parte II: PREGUNTAS CORTAS. 10 %.

Sea la siguiente gramática:

$$
\begin{array}{lcl}
A & \to & B\ C\ A \mid a \\
B & \to & C\ A\ B \mid b \mid \lambda \\
C & \to & A\ B\ C \mid c \mid \lambda
\end{array}
$$

Se pide:

1. Calcular el conjunto PRIMERO para cada símbolo.
2. Calcular el conjunto SIGUIENTE para cada símbolo.
3. Calcular la función *Predict* para cada regla.
4. Justificar, sin construir tabla de análisis, si esta gramática puede ser o no LL(1).

Parte III: PROBLEMA. 60 %.

La siguiente gramática G permite expresar parcialmente las sentencias `if-then` e `if-then-else` de un lenguaje de programación ($V_T = \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{print}, \texttt{id}, \texttt{==}, \texttt{num}, \texttt{str}\}$ y $V_N = \{S, C, E\}$):

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

1. *(1 punto)* Comprueba si la gramática G es LL(1), calculando los conjuntos PRIMERO y SIGUIENTE, así como la tabla de análisis.

2. *(1 punto)* Con la tabla obtenida en el apartado anterior, analiza la entrada

   `if id num then else print str`

   usando tratamieto de errores en *modo pánico*, y explicando el significado de cada acción realizada en dicha recuperación de errores.

3. *(1.5 puntos)* Comprueba si la gramática G es SLR(1).

4. *(0.5 puntos)* Indica de forma justificada si la gramática es LR(1) y/o LALR(1), sin calcular ninguna colección de ítems adicional.

5. *(0.5 puntos)* Demuestra que la gramática G es ambigua o bien que no lo es ¿Permite G anidamiento en las sentencias `if-then-else`?

6. *(1.5 puntos)*

   a) Realiza una definición dirigida por la sintaxis (DDS) que permita asignar al símbolo inicial de la gramática la cadena literal que se imprimiría en cada caso, así como el nivel de anidamiento al que corresponde dicha cadena[1]. Suponemos que cada identificador tiene almacenado un valor numérico en la tabla de símbolos, que sería el valor asignado a ese identificador en una sentencia previa del programa.

   b) Decora el árbol de análisis para la siguiente entrada, suponiendo que en la tabla de símbolos `a` tiene almacenado el valor `1` y `b` el valor `2`:

   `if a==0 then print ‘‘a es 0’’ else if b==1 print ‘‘a no es 0; b es 1’’ else print ‘‘a no es 0; b no es 1’’`

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

[1] Entendemos que la sentencia `if` más exterior tendría nivel de anidamiento `0`, la sentencia que cuelque de ella, nivel `1`, y así sucesivamente.