**PREGUNTAS TEST COMPILADORES - TEMA 2 (2° Grado en Informática)**

FACULTAD DE INFORMÁTICA

1. El **análisis de léxico**:

   *a*) Debe realizarse necesariamente de forma independiente al *sintáctico*, puesto que las palabras de los lenguajes de programación son generadas por *gramáticas regulares* y las frases por *gramáticas libres de contexto.*

   *b*) Podría realizarse de forma conjunta al *análisis sintáctico*, añadiendo reglas de producción para generar las palabras del lenguaje, puesto que los *lenguajes regulares* están incluidos en los *libres de contexto.*

   *c*) Se implementa como una función a la que llama el *analizador sintáctico* puesto que las herramientas que se usan para implementar el *análisis sintáctico* no tendrían potencia suficiente para simular *autómatas finitos.*

2. El **análisis de léxico**:

   *a*) Suele funcionar como un módulo independiente del compilador, devolviendo un fichero con la lista de tokens requeridos en el análisis sintáctico.

   *b*) Suele estar integrado en el compilador, mediante una rutina que devuelve un token cada vez que lo solicita el analizador sintáctico.

   *c*) Se resuelve usando *autómatas linealmente acotados* para reconocer las expresiones regulares y devolver los códigos de token.

3. Elige la frase correcta acerca del **análisis de léxico**:

   *a*) Suele ser una función a la que llama el *analizador sintáctico* cada vez que necesita un token, aunque podría no realizarse de forma explícita y dejar el reconocimiento de palabras como parte del análisis sintáctico.

   *b*) Suele generar un fichero explícito de tokens que constituye la entrada del *analizador sintáctico.*

   *c*) Se realiza mediante la simulación de *autómatas de pila*, que proporcionan la potencia suficiente para las tareas de E/S.

4. El análisis de léxico:

   *a*) consiste en la implementación de un autómata de pila.

   *b*) consiste en la implementación de un autómata finito.

   *c*) consiste en la implementación de un autómata linealmente acotado.

5. Elige la frase correcta acerca del análisis de léxico:

   a) Puede crear entradas de identificadores en la tabla de símbolos, aunque se suele dejar esta tarea al *analizador sintáctico.*

   b) Su única misión es agrupar los caracteres del programa fuente en *lexemas* y pasar la secuencia de tokens al *analizador sintáctico.* El *analizador sintáctico* se encargará de otras tareas adicionales como eliminar comentarios y caracteres de espaciado.

   c) Nunca detecta errores de compilación. Los errores se detectan a partir de la fase de *análisis sintáctico.*

6. En la siguiente gramática libre de contexto que genera un lenguaje compuesto por números:

$$\begin{aligned} N &\rightarrow D'\ N \mid D \\ D &\rightarrow 0 \mid 1 \mid ... \mid 9 \\ D' &\rightarrow 1 \mid ... \mid 9 \end{aligned}$$

   *a*) los tokens del lenguaje serían los diez dígitos y, por tanto, no necesitarían atributo para identificar su lexema.

   *b*) el único token asociado a la gramática sería el token NUM, que necesitaría un atributo para almacenar su lexema, en caso de necesitarlo.

   *c*) no hay tokens definidos.

7. Dadas las tres expresiones regulares [^hoy], [^"hoy"] y [^h][^o][^y], escritas en formato *flex*:

   *a*) las tres describen el conjunto de todas las palabras de tres caracteres excepto la palabra *hoy.*

   *b*) sólo la tercera describe el conjunto de todas las palabras de tres caracteres excepto la palabra *hoy.*

   *c*) ninguna de las tres describe el conjunto de todas las palabras de tres caracteres excepto la palabra *hoy.*

8. A partir de una expresión regular:

   *a*) No puede obtenerse directamente un AFD, sino que hay que convertirla primero en AFND y posteriormente convertir este AFND al AFD equivalente mediante la *construcción de subconjuntos*.

   *b*) Podemos obtener el AFND, aunque éste no puede simularse directamente, sino que habría que convertirlo previamente a AFD.

   *c*) Podemos obtener directamente el AFD o el AFND y simular cualquiera de los dos.

9. Dada una especificación léxica de Flex con la siguiente lista de expresiones regulares:
   $(00)*$
   $01+$
   $10+$
   ¿Cuál de las siguientes cadenas no se podría procesar satisfactoriamente?

   *a*) 0111110

   *b*) 0001101

   *c*) 01100100

10. Para que el analizador léxico distinga entre los operadores de asignación '=' y de comparación '==':

    *a*) Necesitaría usar una condición de arranque al estilo de flex.

    *b*) Basta con que haga uso de un carácter de anticipación.

    *c*) Es necesario usar un buffer dividido en dos mitades.

11. Dada la siguiente especificación léxica, usada para generar un analizador con Flex:
    $a(ba)*$
    $b*(ab)*$
    $abd$
    $d+$
    ¿cuál de las siguientes afirmaciones es correcta?

    *a*) La cadena *ababddababa* se descompondrá en tokens con lexemas *ab*, *abd*, *d* y *ababa*.

    *b*) La cadena *ababdddd* se descompondrá en tokens con lexemas *abab* y *dddd*.

    *c*) La cadena *dddabbabab* se descompondrá en tokens con lexemas *ddd*, *a* y *bbabab*.

12. Decir cuál de las siguientes afirmaciones es **falsa**:

    *a*) Los comentarios anidados pueden generarse con una gramática regular, aunque no suelen permitirse por cuestiones de eficiencia.

    *b*) Los comentarios anidados no pueden generarse con una gramática regular, por eso no suelen permitirse.

    *c*) Los comentarios anidados podrían reconocerse con la herramienta flex usando condiciones de arranque o de contexto.

**PREGUNTAS CORTAS.**

1. Dada la gramática G con el siguiente conjunto de producciones:

$$\begin{array}{lcl} T & \rightarrow & B\ C \\ B & \rightarrow & int \mid float \\ C & \rightarrow & [\ num\ ]\ C \ \mid \lambda \end{array}$$

   que genera el lenguaje para declarar arrays en C, enumerar los tokens de G, indicando cuales tendrían atributo asociado en caso de que usáramos la gramática para realizar un compilador. Finalmente, dada la entrada

```
int[45][2]
```

   indicar qué información proporcionaría el analizador léxico al sintáctico.

2. La siguiente gramática:

   $baseNum \rightarrow num\ basechar$
   $basechar \rightarrow$ `octal`
   $basechar \rightarrow$ `decimal`
   $num \rightarrow num\ dig$
   $num \rightarrow dig$
   $dig \rightarrow 0$
   $dig \rightarrow 1$
   $dig \rightarrow 2$
   $dig \rightarrow 3$
   $dig \rightarrow 4$
   $dig \rightarrow 5$
   $dig \rightarrow 6$
   $dig \rightarrow 7$
   $dig \rightarrow 8$
   $dig \rightarrow 9$

   es una gramática libre de contexto, expresada en notación BNF

   *a*) ¿Crees que existe una gramática regular equivalente? Justifica la respuesta.

   *b*) ¿Cuáles son los símbolos terminales o tokens de esta gramática?

   *c*) Si estas reglas formaran parte de una gramática más general, que implementara una calculadora, por ejemplo, ¿descargarías parte del reconocimiento de los números en octal y decimal en el analizador léxico en lugar de usar estas reglas? En ese caso, ¿qué símbolos terminales o tokens definirías para que fueran devueltos por el analizador léxico?