EXAMEN DE COMPILADORES (2° **Grado en Informática, final junio-2011**)

FACULTAD DE INFORMÁTICA

Apellidos, nombre: **GRUPO:**
D.N.I.:
Este enunciado y todos los folios usados deben entregarse al salir

Parte I: PREGUNTAS TIPO TEST. 30 %.
Cada respuesta correcta vale 0.2 puntos.
Cada dos respuestas incorrectas anulan una correcta.

1. La sintaxis de los lenguajes de programación...
   a) ...se analiza con *autómatas de pila*, puesto que suelen ser generados por *gramáticas libres de contexto.*
   b) ...se analiza con *autómatas linealmente acotados*, puesto que suelen ser generados por *gramáticas sensibles al contexto.*
   c) ...se analiza con *autómatas de pila*, aunque suelen ser *sensibles al contexto.*
2. 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.*
3. Realizar un compilador...
   a) ...que siempre garantice la generación del código más óptimo es posible, aunque generalmente no merece la pena.
   b) ...que siempre garantice la generación del código más óptimo es imposible, puesto que su realización entra dentro de los llamados *problemas NP-completos.*
   c) ...que genere código óptimo es imposible, puesto que aún no se conocen las técnicas de optimización necesarias.
4. Una *máquina virtual*...
   a) ...es un programa que sirve para que funcionen algunos compiladores que generan código intermedio, con el fin de aumentar la velocidad en el proceso de traducción.
   b) ...es un intérprete de un lenguaje de bajo nivel que funciona, salvo en la velocidad, como una máquina real para dicho lenguaje.
   c) ...se usa en aquellos lenguajes que jamás podrían ser analizados por una máquina real, funcionando de forma semejante a dicha máquina.
5. Elige, entre las siguientes, la frase que consideres más adecuada para definir el proceso de *arranque*:
   a) Se produce cuando en un procesador de lenguaje coinciden el *lenguaje de implementación* y el *lenguaje fuente.*
   b) Se produce cuando en un procesador de lenguaje coinciden el *lenguaje de implementación* y el *lenguaje destino.*
   c) Se produce cuando en un procesador de lenguaje el *lenguaje destino* es un *lenguaje intermedio.*
6. Un *compilador portable*
   a) ...se consigue escribiéndolo en un lenguaje de alto nivel y compilándolo cada vez que se quiera cambiar de máquina.
   b) ...puede conseguirse implementando un intérprete para un lenguaje intermedio adecuado.
   c) ...no puede existir, puesto que al cambiar de máquina seguirá generando código para la máquina original.
7. Las técnicas de *recuperación de errores* en el proceso de traducción...
   a) ...tienen como objetivo obtener un código objeto sin los errores introducidos por el usuario.
   b) ...pretenden exclusivamente informar al usuario de los errores de la forma más precisa posible.
   c) ...intentan localizar el máximo posible de errores.

8. Una *gramática con λ-reglas*...

   a) ...puede ser LL y LR.

   b) ...puede ser LL pero no LR.

   c) ...puede ser LR pero no LL.

9. El análisis sintáctico LL(1) consiste...

   a) ...en una técnica ascendente dónde se realiza la derivación más a la izquierda.

   b) ...en una técnica descendente dónde se realiza la derivación más a la derecha.

   c) ...en una técnica descendente dónde se realiza la derivación más a la izquierda.

10. Elige la opción correcta, suponiendo que consideramos un único símbolo de anticipación para cada método:

    a) Una *gramática LALR* es *SLR* aunque no tenemos seguridad de que sea *LR-canónica*.

    b) Una *gramática LALR* es *LR-canónica* aunque no tenemos seguridad de que sea *SLR*.

    c) Una *gramática LR-canónica* es *SLR* y *LALR*.

11. Si a partir del automáta de una gramática LR-Canónica construimos el automáta de la gramática LALR, al construir la tabla...

    a) ...pueden aparecer conflictos shift/reduce y reduce/reduce.

    b) ...sólo pueden aparecer conflictos reduce/reduce.

    c) ...sólo pueden aparecer conflictos shift/reduce.

12. Elige la opción correcta:

    a) Una *gramática L-atribuida* es también *S-atribuida* y permite hacer una evaluación de atributos al tiempo que se realiza un análisis LR.

    b) Una *gramática S-atribuida* es también *L-atribuida* y permite hacer una evaluación de atributos al tiempo que se realiza un análisis LL.

    c) Siempre que una gramática atribuida sea *no circular*, se puede realizar una evaluación de sus atributos.

13. La *tabla de símbolos*...

    a) ...almacena información útil acerca de los identificadores de un lenguaje de programación, siendo útil exclusivamente en la etapa de análisis del proceso de compilación.

    b) ...es una estructura de datos que se usa a lo largo de todo el proceso de compilación y que sirve, entre otras cosas, para poder controlar las restricciones contextuales de un lenguaje de programación.

    c) ...es una estructura de datos donde se almacena el código intermedio generado por el compilador.

14. La *comprobación de tipos*...

    a) ...podría no ser idéntica en implementaciones diferentes del mismo lenguaje de programación.

    b) ...tiene que realizarse necesariamente en tiempo de compilación, puesto que en otro caso podrían producirse errores semánticos en la ejecución de un programa.

    c) ...en tiempo de compilación es innecesaria en los lenguajes fuertemente tipificados.

15. Indica cual es la afirmación es correcta:

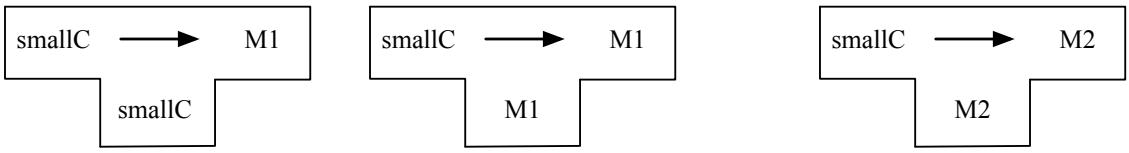
    a) Un árbol sintáctico puede usarse para la generación de código.

    b) Un GDA es una representación de bajo nivel.

    c) La anchura de un tipo almacena la dirección absoluta de una variable.

Parte II: PREGUNTAS CORTAS. 10 %.

1. *(0.5 puntos)* Se dispone de un compilador del lenguaje fuente *smallC* para la máquina *M1*. Dicho compilador ha sido escrito en el lenguaje de implementación *smallC* (configuración de partida). Se pretende crear una versión del compilador de *smallC* para la máquina *M2* (configuración objetivo). Para hacerlo no se dispone de ningún otro compilador distinto al indicado en la configuración de partida. ¿Cuál es el proceso más adecuado para llegar a la configuración objetivo? Indica las etapas de dicho proceso.



a) Configuración de partida b) Configuración objetivo

2. *(0.5 puntos)* La siguiente gramática describe sentencias con asignaciones múltiples. ¿Qué modificaciones harías en la gramática para llegar a una equivalente que pudiese ser reconocida con un analizador LL(1)?

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

Parte III: PROBLEMAS. 60 %

Supongamos que se desea formalizar un lenguaje de matrices numéricas en el que, por ejemplo, una matriz de dos filas y tres columnas como

$$\begin{pmatrix} -1 & 3 & 7 \\ 4 & 6 & 0 \end{pmatrix}$$

quedaría representada de la siguiente forma:

$$(\ -1\ 3\ 7\ ;\ 4\ 6\ 0\ ;\ )$$

La siguiente gramática, G, con $V_T = \{(,), num, ;\}$, $V_N = \{MATRIZ, FILA, FILAS, NUMEROS\}$, símbolo inicial $MATRIZ$ y el siguiente conjunto P de producciones:

$$\begin{array}{lcl} MATRIZ & \rightarrow & (\ FILA\ FILAS\ ) \\ FILA & \rightarrow & num\ NUMEROS\ ; \\ FILAS & \rightarrow & FILA\ FILAS \\ & | & \lambda \\ NUMEROS & \rightarrow & num\ NUMEROS \\ & | & \lambda \end{array}$$

sirve para generar matrices con el formato anterior.
Realiza los siguientes ejercicios:

1. *(1.5 puntos)* Comprueba si se trata de una gramática LL(1) calculando los conjuntos PRIMERO y SIGUIENTE para cada símbolo no terminal, los conjuntos *predict* para cada regla y la tabla de análisis.

2. *(0.5 puntos)* Simular el comportamiento del algoritmo de análisis LL para la cadena w ≡ (num num ; ;), realizando la recuperación de errores en modo pánico en caso de error.

3. *(1.75 puntos)* Construir la colección LR(1) para G y la tabla LR-canónica. Indicar si G es una gramática LR-canónica justificando la respuesta.

4. *(0.75 puntos)* Indicar si G es una gramática LALR y/o SLR justificando la respuesta, y sin calcular ninguna colección de items adicional.

5. *(1.5 puntos)* Dar una *definición dirigida por la sintaxis* para añadir la restricción de contexto que permita controlar que todas las filas de la matriz tengan el mismo número de elementos, de manera que se llame a una función *error* en el caso de que alguna fila no tenga el mismo número de elementos que la primera. Para esto:

   - indicar el número y tipo de atributos asociado a cada símbolo de G.
   - asociar a cada regla de producción de G las acciones semánticas necesarias.
   - decorar el árbol sintáctico correspondiente a la cadena $w \equiv (\ -1\ 3\ 7\ ;\ 4\ 6\ 0\ ;\ )$.
   - indicar si G es S-atribuida y/o L-atribuida justificando la respuesta.