EXAMEN DE COMPILADORES (2° Grado en Informática, final junio-2018)

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. Cuando compilamos un compilador con otro:
   a) El lenguaje fuente de ambos tiene que coincidir.
   b) El lenguaje de implementación de ambos tiene que coincidir.
   c) El lenguaje destino de ambos no tiene que coincidir necesariamente.
2. La semántica de un lenguaje de programación:
   a) No está incluida en la descripción BNF de la gramática, puesto que en una traducción automática el significado de las frases es irrelevante.
   b) No está incluida en la descripción BNF de la gramática, pero hay que tenerla en cuenta en la implementación del compilador.
   c) Aparece en la decripción BNF de la gramática, de forma implícita.
3. ¿Cuál de las siguientes gramáticas genera el lenguaje $L = \{\ a^n b^m \mid 1 \leq m \leq n \leq 3m\ \}$
   a)
   $$S \rightarrow aSb \mid aaSb \mid aaaSb \mid \lambda$$
   b)
   $$\begin{array}{lcl} S & \rightarrow & aXb \mid aaXb \mid aaaXb \\ X & \rightarrow & \lambda \mid aXb \mid aaXb \mid aaaXb \end{array}$$
   c)
   $$\begin{array}{lcl} S & \rightarrow & aXb \\ X & \rightarrow & \lambda \mid aaXb \mid aaaXb \end{array}$$
4. Sólo una de estas tres expresiones regulares *flex* reconoce de forma completa la cadena `¿C++ o Java?`. Indicar cuál:
   a)
   ```
   cadena  (a-zA-Z|+| )

   %%

   "¿"{cadena}*"?"
   ```
   b)
   ```
   cadena  [a-zA-Z|\+|\?|¿]

   %%

   {cadena}+
   ```
   c)
   ```
   ¿C[+]+[?a-z A-Z]*
   ```
5. En el analizador léxico de la práctica para el lenguaje *miniC*, realizado con Flex:
   a) Todas las expresiones regulares reconocen lexemas asociados a algún token.
   b) Cada token tiene un atributo asociado que hay que pasar a Bison a través de la variable `yylval`.
   c) Cada token tiene un lexema asociado que se almacena total o parcialmentte en la variable `yytext`.
6. La siguiente gramática:
   $$S \rightarrow a \mid b\ S\ c\ S\ b \mid \lambda$$
   a) Es LL(1).
   b) No es LL(1) pero sí SLR(1).
   c) No es LR(1).

7. La gramática anterior:

   a) Es λ-libre puesto que la única λ-regla tiene al símbolo inicial en su parte izquierda.

   b) Es propia pues, aunque tiene reglas unitarias, no tiene ciclos.

   c) No es propia.

8. Dada la sentencia `bacb` generada por la gramática anterior, el pivote[1] es:

   a) `bac`$\underline{\lambda}$`b`.

   b) `b`$\underline{\text{a}}$`cb`.

   c) $\underline{\text{b}}$`acb`.

9. Consideramos la gramática G siguiente:

$$\begin{array}{lll} S & \rightarrow (\ L \ ) \mid id \\ L & \rightarrow S \ L' \\ L' & \rightarrow, \ S \ L' \mid \lambda \end{array}$$

   a) G es LL(1), $predict(L' \rightarrow \lambda) = \{), \$\}$, y la tabla de análisis es

NO TERM	**TERMINAL**				
	(	)	,	*id*	$
S	$S \rightarrow (\ L \ )$			$S \rightarrow id$	
L	$L \rightarrow S \ L'$			$L \rightarrow S \ L'$	
L'		$L' \rightarrow \lambda$	$L' \rightarrow, \ S \ L'$		$L' \rightarrow \lambda$

   b) G es LL(1), $predict(L' \rightarrow \lambda) = \{)\}$, y la tabla de análisis es

NO TERM	**TERMINAL**				
	(	)	,	*id*	$
S	$S \rightarrow (\ L \ )$			$S \rightarrow id$	
L	$L \rightarrow S \ L'$			$L \rightarrow S \ L'$	
L'		$L' \rightarrow \lambda$	$L' \rightarrow, \ S \ L'$		

   c) G no es LL(1) puesto que $predict(L' \rightarrow, \ S \ L') \ \cap \ predict(L' \rightarrow \lambda) = \{, \}$.

10. Con respecto a la simulación desdendente predictiva, señalar la afirmación $\underline{\text{incorrecta}}$:

    a) Si en la pila de análisis aparece `$)L'` y en la entrada `,id)$`, en el siguiente paso de cálculo, la pila contendría `$)L'S,` y la entrada `,id)$`.

    b) Si en la pila de análisis aparece `$)L'` y en la entrada `id)$`, en el siguiente paso de cálculo, se daría un mensaje de error, la pila contendría `$)L'S` y la entrada `id)$`, suponiendo una recuperación de errores en modo pánico.

    c) Si en la pila de análisis aparece `$)L'` y en la entrada `id)$`, en el siguiente paso de cálculo, se daría un mensaje de error, la pila contendría `$)` y la entrada `)$`, suponiendo una recuperación de errores en modo pánico.

11. En la gramática anterior ¿cuáles serían las tres primeras reducciones si se realiza un análisis ascendente de la entrada `(id,id)`?

    a) $S \rightarrow id$, $S \rightarrow id$, $L' \rightarrow \lambda$

    b) $S \rightarrow id$, $L \rightarrow S \ L'$, $S \rightarrow (\ L \ )$

    c) $S \rightarrow id$, $S \rightarrow id$, $L' \rightarrow, \ S \ L'$

12. La gramática del lenguaje *miniC* de las prácticas:

    a) Es LALR, pues no podría procesarse en otro caso con Bison.

    b) No es LALR, ni siquiera LR-Canónica.

    c) No es SLR pues presenta conflictos en expresiones aritméticas y sentencias `if/if-else`. Los comandos `%left` de Bison, en cambio, hace que sea LALR.

[1] λ representa la cadena vacía.

13. Dado el siguiente fragmento de un fichero con formato Bison:

```
%{
int cs=0;
%}
%token  INICIO FIN CAB COL
%%
inicial : { cs++; } INICIO cabecera cola FIN {if (cs) printf("%d\n",$3);else printf("%d\n",$4);};
cabecera : CAB    {$$=2017;cs--;}
           |      {$$=2018;cs--;};
cola : COL         {$$=2019;cs++;}
           |       {$$=2020;cs++;};
```

La salida del programa en C generado a partir de él, ante la entrada `INICIO FIN` será:

a) 2017 o 2019

b) 2018

c) 2020

14. Dada la siguiente gramática G con $V_T = \{0, 1\}$ y $V_N = \{S, A, B\}$, siendo P:

$$\begin{array}{ll} S & \rightarrow A\ B \mid 0\ S\ 1 \mid A \mid B \mid \lambda \\ A & \rightarrow 0\ A\ B \mid 0\ B \mid 0\ A \mid 0 \\ B & \rightarrow B\ 1 \mid 1 \end{array}$$

podemos afirmar:

a) La gramática es recursiva por la izquierda, aunque no ambigua.

b) La gramática es LR(1).

c) El `0` no es seguidor de ningún no terminal.

15. Dada la siguiente gramática G con $V_T = \{a, x\}$ y $V_N = \{S, X\}$, siendo P:

$$\begin{array}{ll} S & \rightarrow X\ S \mid a \\ X & \rightarrow S\ X \mid x \end{array}$$

y la siguiente DDS:

Regla de producción	**Regla semántica**
$S \rightarrow X\ S_1$	S.val = f(X.val);S_1.val = f(X.val);
$S \rightarrow a$	S.val = g(a.val);
$X \rightarrow S\ X_1$	X.val = h(S.val,X_1.val);S.val = l(X_1.val);
$X \rightarrow x$	X.val = f(x.val);

Podemos afirmar:

a) La gramática es S-atribuida.

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

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

Parte II: PROBLEMA. 70 %.
Sea G la gramática con $V_T = \{L, C, T\}$ y $V_N = \{[,], ,, \texttt{id}\}$, con P:

$$\begin{array}{ll} L & \rightarrow [\ C\ ] \\ C & \rightarrow \lambda \mid T \\ T & \rightarrow T\ ,\ T \mid id \mid L \end{array}$$

G genera un lenguaje que representa de forma parcial listas de Prolog. Hay que tener en cuenta que en las listas, la coma es asociativa por la izquierda.

1. *(1.25 puntos)* Decir si la gramática G es LL(1). Indicar todos los problemas que puedan impedir que lo sea, si los hay. En este caso, transformar la gramática, intentando conseguir una gramática equivalente LL(1).

2. *(1.25 puntos)* Con la gramática obtenida en el apartado anterior, G', calcular los conjuntos PRIMERO, SIGUIENTE, PREDICT, tabla LL(1) y razonar si G' es LL(1).

3. *(2.5 puntos)* Calcular la colección LR-canónica o LR(1) para la gramática G original. Calcular la tabla LALR(1) y deducir si la gramática es LALR(1). En caso de que aparezcan conflictos en la tabla LALR(1), resolverlos. Indicar también si la gramática es LR(1) y SLR(1).

4. *(0.75 puntos)* Con la tabla del apartado anterior, simular con tratamiento de errores la entrada `[a,[b[]]`.

5. *(1.25 puntos)* Implementa una DDS que obtenga una única lista con los elementos de la lista de entrada en orden inverso. Decorar el árbol para la entrada `[a,[b,c],d]`, que debe retornar `[d,c,b,a]`. Indica si es S-atribuida y/o L-atribuida.