EXAMEN DE COMPILADORES (2° **Grado en Informática, final enero-2026)**

Facultad Informática Universidad Murcia

Apellidos, nombre: **DNI:**

Instrucciones: Este enunciado y todos los folios usados deben entregarse al salir **Grupo:**

Señala a qué parte te presentas:	TEORÍA	PRÁCTICAS

EXAMEN DE PRÁCTICAS

Escribe en esta tabla tus respuestas:

1	2	3	4	5

1. Supongamos que se emplea la siguiente especificación de un analizador léxico con Flex:

```
%{
    #include <stdio.h>
%}
D       [0-9]
L       [a-zA-Z_]
%%
{L}({L}|{D}){1,9} { printf("ER1: %s\n", yytext); }
.                 { printf("ER2: %s\n", yytext); }
({L}|{D})*        { printf("ER3: %s\n", yytext); }
%%
```

y que la entrada es la cadena `a1234567890`. ¿Qué salida producirá el analizador?

a) `ER1: a123456789`
`ER1: 0`

b) `ER3: a1234567890`

c) `ER1: a123456789`
`ER2: 0`

2. Supongamos que se está implementando la generación de código del `if` de miniC, y se emplea la siguiente regla de bison parcialmente incompleta:

```
1  statement : IF "(" expression ")" statement {
2      if (!errores) {
3          $$ = $3;
4          Operacion oper;
5          char *l;
6          oper.op = "beqz";
7          oper.res = /* TODO */;
8          oper.arg1 = l = obtener_etiqueta();
9          oper.arg2 = NULL;
10         insertaLC($$,finalLC($3),oper);
11         concatenaLC( /* TODO */ );
12         oper.op = "etiq";
13         oper.res = l;
14         oper. arg1 = oper.arg2 = NULL;
15         insertaLC($$,finalLC($$),oper);
16         liberar_registro(recuperaResLC($3));
17         liberaLC($5);
18     }
19 }
```

Indica la opción que especifica correctamente lo que hay que escribir en los `/* TODO */` de las líneas 7 y 11:

a) Línea 7: `recuperaResLC($5)`
Línea 11: `$$, $5`

b) Línea 7: `recuperaResLC($3)`
Línea 11: `"_", oper.res`

c) Línea 7: `recuperaResLC($3)`
Línea 11: `$$, $5`

3. Teniendo en cuenta la especificación de la gramática básica (sin ampliaciones) de miniC indicada al final de este examen, selecciona cuál de las siguientes es una expresión regular correcta para llevar a cabo el tratamiento de errores en modo pánico del analizador léxico:

   a) `[^ \n\r\ta-zA-Z_0-9+\-*/=;,(){}?:"]+`

   b) `[^ \n\r\ta-zA-Z_0-9+*/=;,(){}?:-]+`

   c) `[^ \n\r\ta-zA-Z_0-9$+-*/=;,(){}?:"]+`

4. Elige la opción <u>incorrecta</u>:

   a) En el compilador de miniC, la traducción asignada a la variable $$ correspondiente a la regla:
   `expression : ID`
   puede formar parte de la traducción asignada posteriormente a $$ en la regla:
   `expression : expression MAS expression`

   b) En el compilador de miniC, la traducción asignada a la variable $$ correspondiente a la regla:
   `expression : expression MAS expression`
   puede formar parte de la traducción asignada posteriormente a $$ en la regla:
   `expression : expression MAS expression`

   c) En el compilador de miniC, la traducción asignada a la variable $$ correspondiente a la regla:
   `expression : ID`
   puede formar parte de la traducción asignada posteriormente a $$ en la regla:
   `var_list : var_list, ID`

5. Supongamos que hacemos una especificación de un analizador sintáctico del lenguaje miniC en bison empleando la siguiente función auxiliar para concatenar listas de código:

```
ListaC concat_lists(ListaC list1, ListaC list2) {
    ListaC codigo = NULL;
    if (!errores) {
        codigo = list1;
        concatenaLC(codigo, list2);
        liberaLC(list2);
    }
    return codigo;
}
```

   donde `errores` es una variable que se incrementa al detectar cualquier tipo de error durante el análisis. ¿Cuál de las siguientes es una implementación correcta de la traducción del no terminal `declarations`?

   a)
```
%type <codigo> const_list declarations
%%
declarations    : declarations VAR  tipo var_list ";"
                  { $$ = concat_lists($1, $4); }
                | declarations CONS tipo const_list ";"
                  { $$ = concat_lists($1, $4); }
                | %empty { $$ = creaLC(); }
                ;
```

   b)
```
%type <codigo> const_list declarations
%%
declarations    : declarations VAR  tipo var_list ";"
                  { $$ = $1; }
                | declarations CONS tipo const_list ";"
                  { $$ = concat_lists($1, $4); }
                | %empty { $$ = creaLC(); }
                ;
```

   c)
```
%type <codigo> const_list declarations
%%
declarations    : declarations VAR  tipo var_list ";"
                  { $$ = creaLC(); }
                | declarations CONS tipo const_list ";"
                  { $$ = concat_lists($1, $4); }
                | %empty { $$ = creaLC(); }
                ;
```

Referencia: sintaxis de miniC.

program	→	*id* `(` `)` { declarations statement_list }
declarations	→	declarations `var` tipo var_list `;`
	|	declarations `const` tipo const_list `;`
	|	λ
tipo	→	`int`
var_list	→	*id*
	|	var_list `,` *id*
const_list	→	*id* `=` expression
	|	const_list `,` *id* `=` expression
statement_list	→	statement_list statement
	|	λ
statement	→	*id* `=` expression `;`
	|	{ statement_list }
	|	`if` `(` expression `)` statement `else` statement
	|	`if` `(` expression `)` statement
	|	`while` `(` expression `)` statement
	|	`print` `(` print_list `)` `;`
	|	`read` `(` read_list `)` `;`
print_list	→	print_item
	|	print_list `,` print_item
print_item	→	expression
	|	*string*
read_list	→	*id*
	|	read_list `,` *id*
expression	→	expression `+` expression
	|	expression `-` expression
	|	expression `*` expression
	|	expression `/` expression
	|	(expression ? expression : expression)
	|	`-` expression
	|	`(` expression `)`
	|	*id*
	|	*num*

Funciones de listaCodigo:

```
/* Crea una lista de código */
ListaC creaLC();
/* Destruye una lista de código */
void liberaLC(ListaC codigo);
/* Inserta una nueva operación en la lista de código, en la posición indicada */
void insertaLC(ListaC codigo, PosicionListaC p, Operacion o);
/* Concatena dos listas de código. La primera lista se modifica para formar el resultado */
void concatenaLC(ListaC codigo1, ListaC codigo2);
/* Posición de final de una lista de código */
PosicionListaC finalLC(ListaC codigo);
/* Recupera el registro resultado de una lista de código */
char * recuperaResLC(ListaC codigo);
```

EXAMEN DE TEORÍA

Parte I: PREGUNTAS TIPO TEST. 40 %. Cada dos respuestas incorrectas anulan una correcta.

Escribe en esta tabla tus respuestas:

1	2	3	4	5	6	7	8	9	10

1. Algunos compiladores realizan operaciones de conversión de tipos de forma automática, como sucede al operar un número entero con un número flotante. ¿En qué fase del compilador se realiza esta conversión?

   a) En el analizador léxico.

   b) En la generación de código intermedio.

   c) En el analizador semántico.

2. De entre las opciones siguientes, indica la que no reúne las condiciones para emplearse como un código intermedio entre el bloque de análisis y el bloque de síntesis de un compilador:

   a) Árbol sintáctico.

   b) Código de tres direcciones.

   c) Secuencia de tokens.

3. Dada la siguiente gramática con $V_T = \{x, y, z\}$ y $V_N = \{A\}$:

$$A \rightarrow AAx \mid AAy \mid z$$

   indica cuál de las siguientes es una gramática equivalente que reúne las condiciones para ser tratada con el método LL(1):

   a) $A \rightarrow zA''$
   $A'' \rightarrow \lambda \mid A'$
   $A' \rightarrow AxA'' \mid AyA''$

   b) $A \rightarrow zA'$
   $A' \rightarrow \lambda \mid AA''$
   $A'' \rightarrow xA' \mid yA'$

   c) $A \rightarrow zA'$
   $A' \rightarrow \lambda \mid AA''$
   $A'' \rightarrow xA \mid yA$

4. Dada la siguiente gramática con $V_T = \{f, (,), ;, a\}$ y $V_N = \{S, A\}$:

$$\begin{aligned} S &\rightarrow f(aA) \\ A &\rightarrow ;aA \mid \lambda \end{aligned}$$

   supongamos que se realiza un análisis descendente predictivo de la cadena $fa;a)\$$. ¿Cuál de las siguientes es la configuración en la que se detectaría el error sintáctico?

   a)

PILA	**ENTRADA**
$\$(aA)$	$a;a)\$$

   b)

PILA	**ENTRADA**
$\$)Aa($	$a;a)\$$

   c)

PILA	**ENTRADA**
$\$)aA($	$a;a)\$$

5. Continuando con la gramática del ejercicio anterior y la validación de la cadena $fa;a)\$$ con el método de análisis descendente predictivo, ¿qué estrategia de tratamiento del error en modo pánico se empleará cuando se llegue a la configuración de error?

   a) Se eliminará el símbolo (de la pila.

   b) Se eliminará A de la pila y se saltará hasta).

   c) La cadena indicada contiene un error irrecuperable con el modo pánico.

6. En relación con las tablas de los métodos de análisis LR(1), LALR(1) y SLR(1), indica la respuesta correcta:

   a) Hay gramáticas en las que las tablas de los tres métodos son exactamente iguales.

   b) Necesariamente la tabla SLR(1) tiene más reducciones que las tablas LR(1) y LALR(1) por realizar esta acción en todos los símbolos de los conjuntos SIGUIENTE de los no terminales.

   c) Para algunas gramáticas, las tablas SLR(1) y LALR(1) pueden tener la misma cantidad de estados, pero la tabla LR(1) siempre tiene un número mayor de estados porque emplea los símbolos de anticipación en las reducciones.

7. Supongamos que se emplea la gramática ambigua de expresiones aritméticas con el método SLR, siendo $V_T = \{a, +, -, *, /\}$ y $V_N = \{E\}$:

$$E \rightarrow E + E \mid E - E \mid E * E \mid E/E \mid a$$

Al analizar la cadena $a + a * a$ se alcanza una entrada de la tabla SLR con un conflicto. Indica qué items intervendrán en el conflicto provocando las acciones de desplazamiento y reducción:

a) $E \rightarrow E + E\cdot$, $E \rightarrow \cdot E * E$

b) $E \rightarrow E + E\cdot$, $E \rightarrow E \cdot * E$

c) $E \rightarrow E \cdot + E$, $E \rightarrow E * E\cdot$

8. Dada la siguiente gramática:

$$\begin{array}{lcl} (1)\ P & \rightarrow & P\ C \\ (2) & \mid & \lambda \\ (3)\ C & \rightarrow & loop\ P\ fin \\ (4) & \mid & print \end{array}$$

a) No es LL(1) ni LR(1) porque no es propia.

b) No es propia pero sí LL(1).

c) No es propia ni LL(1).

9. Dada la siguiente gramática, para reconocer con un Analizador Descendente Predictivo Recursivo la cadena `loop fin`:

$$\begin{array}{lcl} (1)\ P & \rightarrow & C\ P \\ (2) & \mid & \lambda \\ (3)\ C & \rightarrow & loop\ P\ fin \\ (4) & \mid & print \end{array}$$

a) No es necesario realizar ninguna llamada al método `P()`.

b) Es necesario realizar dos llamadas al método `P()`.

c) Es necesario realizar tres llamadas al método `P()`.

10. Dada la siguiente gramática:

$$\begin{array}{lcl} (1)\ A & \rightarrow & B\ A \\ (2) & \mid & \lambda \\ (3)\ B & \rightarrow & a\ A\ c \\ (4) & \mid & b \end{array}$$

y un analizador SLR(1) para reconocer cadenas derivadas de esa gramática. Elige la opción <u>incorrecta</u>:

a) Si `0a3B2A5` es una configuración correcta de la pila y el siguiente símbolo en la entrada es `a` o `b`, se producirá necesariamente un error.

b) Si `0a3B2A5` es una configuración correcta de la pila y el siguiente símbolo en la entrada es `c` o $, la acción será r1.

c) Si `0a3A6` es una configuración correcta de la pila y el siguiente símbolo en la entrada es `c`, la acción será r2.

Parte II: EJERCICIOS. 60 %.
Dada la siguiente gramática G, con $V_T = \{loop, fin, print\}$, $V_N = \{P, C\}$, y el conjunto de reglas:

$$\begin{array}{lcl} (1)\ P & \rightarrow & C\ P \\ (2) & | & \lambda \\ (3)\ C & \rightarrow & loop\ P\ fin \\ (4) & | & print \end{array}$$

1. *(1 punto)* Indicar las propiedades que, de partida, implican que una gramática no sea LL(1). Decir si G cumple alguna de esas propiedades. En caso de ser así, especificar cuáles y realizar las transformaciones para obtener una gramática G' equivalente a G y que se pueda analizar con el método LL(1). Si no es así, contestar los ejercicios 2, 3 y 4 considerando la gramática G.

2. *(0,75 puntos)* Calcular los conjuntos PRIMERO, SIGUIENTE y PREDICT de los no terminales de G o G' según hayas contestado el ejercicio 1.

3. *(0,5 puntos)* Crear la tabla de análisis LL para G o G' según hayas contestado el ejercicio 1.

4. *(0,25 puntos)* Razonar si G o G' (según hayas contestado el ejercicio 1) es LL(1).

5. *(1 punto)* Completar la colección LR(1) siguiente para la gramática G calculando los conjuntos I_0 e I_3 según las transiciones que se indican. Calcular, además, las transiciones que faltan, es decir, las que conducen a estados previamente calculados.

$I_0 = \{\ \}$

$I_1 = \text{GOTO}(I_0, P) = \{\ [P' \rightarrow P\ \cdot,\ \$\ ]\}$

$I_2 = \text{GOTO}(I_0, C) = \{\ [P \rightarrow C \cdot P,\ \$\ ]$
$[P \rightarrow \cdot C\ P\ ,\ \$\ ]$
$[P \rightarrow \lambda\ \cdot,\ \$\ ]$
$[C \rightarrow \cdot loop\ P\ fin,\ print\ \$\ loop]$
$[C \rightarrow \cdot print,\ print\ \$\ loop]\}$

$I_3 = \text{GOTO}(I_0, loop) = \{\ \}$

$I_4 = \text{GOTO}(I_0, print\ ) = \{[C \rightarrow print\ \cdot,\ print\ \$\ loop]\}$

$I_5 = \text{GOTO}(I_2, P) = \{[P \rightarrow C\ P\ \cdot,\ \$\ ]\}$

$I_6 = \text{GOTO}(I_3, P) = \{[C \rightarrow loop\ P\ \cdot fin,\ print\ \$\ loop]\}$

$I_7 = \text{GOTO}(I_3, C) = \{[P \rightarrow C\ \cdot P,\ fin]$
$[P \rightarrow \cdot C\ P\ ,\ fin]$
$[P \rightarrow \lambda\ \cdot,\ fin]$
$[C \rightarrow \cdot loop\ P\ fin,\ print\ loop\ fin]$
$[C \rightarrow \cdot print,\ print\ loop\ fin]\}$

$I_8 = \text{GOTO}(I_3, loop) = \{[C \rightarrow loop \cdot P\ fin,\ print\ loop\ fin]$
$[P \rightarrow \cdot C\ P\ ,\ fin]$
$[P \rightarrow \lambda\ \cdot,\ fin]$
$[C \rightarrow \cdot loop\ P\ fin,\ print\ loop\ fin]$
$[C \rightarrow \cdot print,\ print\ loop\ fin]\}$

$I_9 = \text{GOTO}(I_3, print) = \{[C \rightarrow print \cdot,\ print\ loop\ fin]\}$

$I_{10} = \text{GOTO}(I_6, fin) = \{[C \rightarrow loop\ P\ fin \cdot,\ print\ \$\ loop]\}$

$I_{11} = \text{GOTO}(I_7, P) = \{[P \rightarrow C\ P \cdot,\ fin]\}$

$I_{12} = \text{GOTO}(I_8, P) = \{[C \rightarrow loop\ P \cdot fin,\ print\ loop\ fin]\}$

$I_{13} = \text{GOTO}(I_{12}, fin) = \{C \rightarrow loop\ P\ fin \cdot,\ print\ loop\ fin]\}$

6. *(0,5 puntos)* Calcular las filas correspondientes a los estados 0 y 8 de la tabla LR-Canónica.

ESTADO	ACCIÓN				IR-A	
	loop	fin	print	$	P	C
0						
8						

7. *(0,5 puntos)* ¿Es la gramática LR-Canónica? Justifica la respuesta.

8. *(0,75 puntos)* Indicar qué conjuntos se agruparían, si los hay, para conseguir la colección LALR ¿Es la gramática LALR? Justifica la respuesta.

9. *(0,75 puntos)* ¿Es la tabla SLR idéntica a la LALR? ¿Es la gramática SLR? Justifica ambas respuestas.