8-04-2019

EXAMEN DE TEORIA DEL PRIMER PARCIAL COMPILADORS

Nombre:

Apellidos:

NIU:

Respuestas al Test de Teoría											
1	a	b	c	d	e	**16**	a	b	c	d	e
2	a	b	c	d	e	**17**	a	b	c	d	e
3	a	b	c	d	e	**18**	a	b	c	d	e
4	a	b	c	d	e	**19**	a	b	c	d	e
5	a	b	c	d	e	**20**	a	b	c	d	e
6	a	b	c	d	e	**21**	a	b	c	d	e
7	a	b	c	d	e	**22**	a	b	c	d	e
8	a	b	c	d	e	**23**	a	b	c	d	e
9	a	b	c	d	e	**24**	a	b	c	d	e
10	a	b	c	d	e	**25**	a	b	c	d	e
11	a	b	c	d	e	**26**	a	b	c	d	e
12	a	b	c	d	e	**27**	a	b	c	d	e
13	a	b	c	d	e	**28**	a	b	c	d	e
14	a	b	c	d	e	**29**	a	b	c	d	e
15	a	b	c	d	e	**30**	a	b	c	d	e

Nota: Las respuestas equivocadas restan ¼ del valor de la pregunta.

ANÁLISIS LEXICOGRÁFICO

1.- ¿Qué operadores se pueden utilizar en las expresiones regulares?
a) AB concatenación, A | B unión, A^+ repetición de una o más veces, A^* repetición de cero o más veces, A-B resta y paréntesis para agrupar operaciones.
b) AB concatenación, A | B unión, A^* repetición de cero o más veces, A-B resta y paréntesis para agrupar operaciones.
c) AB concatenación, A | B unión, [A] opcional, A^+ repetición de una o más veces, A^* repetición de cero o más veces, A-B resta y paréntesis para agrupar operaciones.
d) AB concatenación, A | B unión, A^+ repetición de una o más veces, A^* repetición de cero o más veces y paréntesis para agrupar operaciones.
e) Ninguna de las anteriores.

2.- Un autómata finito determinista acepta una secuencia de caracteres cuando
a) Cuando llega a un estado final.
b) Cuando se queda en un estado final al finalizar la lectura de la secuencia de símbolos.
c) Cuando están definidas todas las transiciones de estado necesarias para que el autómata pueda leer toda la secuencia de caracteres.
d) Cuando sale de un estado final al leer el último carácter de la secuencia.
e) Ninguna de las anteriores.

3.- El algoritmo que obtiene el autómata que acepta el lenguaje generado por una expresión regular realiza los siguientes pasos:
a) Expresión regular, autómata finito determinista, autómata finito no determinista.
b) Expresión regular, autómata finito no determinista.
c) Expresión regular, autómata finito no determinista, autómata finito determinista.
d) Expresión regular, autómata finito determinista, tabla de transiciones.
e) Ninguna de las anteriores.

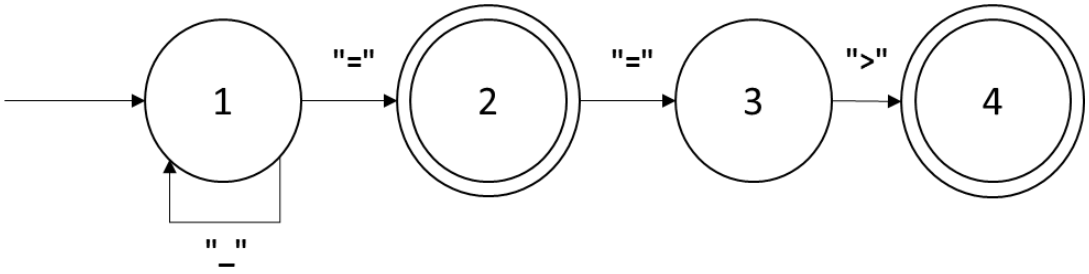
4.- ¿Qué expresiones regulares se necesitan para crea un scanner?
a) Una expresión regular para cada símbolo y una expresión regular única para todos los separadores.
b) Una expresión regular para cada símbolo.
c) Una expresión regular para cada símbolo y una expresión regular para cada tipo de separador.
d) Una única expresión regular para todos los símbolos.
e) Ninguna de las anteriores.

5.- Un scanner…
a) En cada llamada lee caracteres del programa hasta completar la lectura de un símbolo.
b) En una llamada lee todos los caracteres del programa y genera la lista todos los símbolos del programa.
c) En una llamada lee un símbolo y los separadores que le siguen.
d) En una llamada lee caracteres hasta llegar al primer carácter de un separador.
e) Ninguna de las anteriores.

6.- El autómata utilizado para implementar un scanner finaliza la lectura de caracteres
a) Al llegar a un estado final.
b) Al salir de un estado final.
c) Cuando no tiene definida una transición para el estado en que está y el carácter que ha leído.
d) Sólo cuando llega al final del fichero y no hay más caracteres que leer.
e) Ninguna de las anteriores.

7.- ¿Cuáles son las transiciones del autómata determinista de a*b(c|d)$^+$ y su conjunto de estados finales?
a) Trans: 1→ a →2, 2→a →2, 2→b→3, 3→c|d→4, 4→c|d→4. Estados: 4.
b) Trans: 1→ a →1, 1→ b →2, 2 →c →3, 2→d →4, 3→c→3, 4→d→4. Estados: 3,4.
c) Trans: 1→a →1, 1→ b→2, 2 → c|d →3, 3→ c|d →3. Estados: 3.
d) Trans: 1→ a →2, 2→ a →2, 2 →b →3, 3→c →4, 3→d→5, 4→c→4, 5→d→5. Estados: 3,4.
e) Ninguna de las anteriores.

8.- Dado el autómata de un scanner que lee los símbolos igual (“=”), implica (“==>”) y que usa el espacio (“_”) para separarlos, ¿Cuántas llamadas hay que hacer al scanner para leer =_===>_==>?



a) 3
b) 4
c) 5
d) 6
e) Ninguna de las anteriores.

9.- ¿Cuál es el número mínimo de estados necesarios para implementar un autómata determinista que lee como separador el espacio y como símbolos identificador (l(l|d)*), la palabra clave SWITCH y el final de secuencia (#)? ¿Cuántos estados serán finales?
a) 9 estados de los cuales 7 finales.
b) 10 estados de los cuales 9 finales.
c) 8 estados de los cuales 7 finales.
d) 9 estados de los cuales 8 finales.
e) Ninguna de las anteriores.

ANÁLISIS SINTÁCTICO

10.- ¿De que tipo son las gramáticas para definir la sintaxis de un lenguaje de programación?
a) regulares.
b) libres del contexto.
c) dependientes del contexto.
d) generales o no restringidas.
e) Ninguna de las anteriores.

11.- ¿Cuáles son las etapas de análisis de un lenguaje de programación?
a) Lexicográfico, sintáctico y semántico.
b) Lexicográfico, y sintáctico.
c) Scanner y parser.
d) Scaner, parser y generación de código.
e) Ninguna de las anteriores.

12.- ¿Cuáles son las condiciones necesarias, pero no suficientes, para que una gramática escrita en BNF sea LL(1)?
a) Que los conjuntos de primeros de las expresiones BNF que forman una unión sean disjuntos.
b) Que el conjunto de primeros de la expresión BNF que hay dentro de una repetición sea disjunto del conjunto de primeros de lo que le sigue.
c) Que el conjunto de primeros de la expresión BNF que hay dentro de un opcional sea disjunto del conjunto de primeros de lo que le sigue.
d) Todas las anteriores.
e) Ninguna de las anteriores.

13.- Tenemos que escribir una gramática BNF para un lenguaje de expresiones formadas por operadores asociativos por la izquierda y con 5 niveles de prioridad. ¿Cuántas reglas necesitaremos si queremos que el árbol sintáctico generado al analizar una expresión muestre la prioridad de las operaciones?
a) 4 reglas para los operadores.
b) 4 reglas para los operadores y una más para los operandos.
c) 5 reglas para los operadores.
d) 5 reglas para los operadores y una más para los operandos.
e) Ninguna de las anteriores.

14.- ¿Qué pasos se han de seguir para comprobar que una gramática es LL(1)?
a) Cálculo de los primeros de los símbolos no terminales, cálculo de los siguientes de los símbolos no terminales y cálculo de los primeros de las producciones.
b) Cálculo de los símbolos anulables, cálculo de los primeros de los símbolos no terminales, cálculo de los siguientes de los símbolos no terminales y cálculo de los primeros de las producciones.
c) Cálculo de los símbolos anulables, cálculo de los primeros de los símbolos no terminales y cálculo de los primeros de las producciones.
d) Cálculo de los símbolos anulables, cálculo de los primeros de los símbolos no terminales, cálculo de los siguientes de las producciones y cálculo de los primeros de las producciones.
e) Ninguna de las anteriores.

15.- ¿Cuándo una gramática es ambigua?
a) Cuando genera un árbol sintáctico para cada secuencia de símbolos.
b) Cuando genera más de una secuencia de derivación para la misma secuencia de símbolos.
c) Cuando a partir de la misma secuencia de derivación genera más de una secuencia de símbolos.
d) Cuando a partir del mismo árbol sintáctico genera más de una secuencia de símbolos.
e) Ninguna de las anteriores.

16.- Dada la gramática siguiente, ¿Cuáles son los símbolos terminales y no terminales?
<Exp>::= <Fac> { (‘+’|’-‘) <Fac> }
<Fac>::= num | id [‘=’ <Exp>] | ‘(‘ <Exp> ‘)
a) Terminales: ‘+’, ‘-’, num, id, ‘=’, ‘(‘,’)’. No terminales: <Exp>, <Fac>.
b) Terminales: <Exp>, <Fac>. No terminales: ‘+’, ‘-’, num, id, ‘=’, ‘(‘,’)’.
c) Terminales: ‘+’, ‘-’, num, id, ‘=’, ‘[’, ‘]’, ‘(‘,’)’ , ‘{‘,’}’. No terminales: <Exp>, <Fac>.
d) Terminales: <Exp>, <Fac>. No terminales: ‘+’, ‘-’, num, id, ‘=’, ‘[’, ‘]’, ‘(‘,’)’ , ‘{‘,’}’.
e) Ninguna de las anteriores.

17.- ¿Cuántas producciones diferentes surgen al transformar las siguientes reglas BNF de la gramática expuesta a continuación?
Rule <Inicial>::= <Exp>
Rule <Exp>::= <factor> {("+"|"-") <factor>} | <factor> [(<Exp>)]**
Rule <factor>::= numero | identificador ["=" <Exp>]
a) 13
b) 14
c) 15
d) 16
e) Ninguna de las anteriores.

18.- ¿Cuáles son los primeros de los símbolos no terminales de la siguiente gramática?
Rule <Inicial>::= <Exp> ";" | do <factor> while <Exp> ";"
Rule <Exp>::= <factor> ({("+"|"-") <factor>} | {*<factor>})
Rule <factor>::= numero | identificador ["=" <Exp>]

a)
P(Inicial)= do, numero, identificador
P(Exp)= numero, identificador
P(Factor)= numero, identificador

b)
P(Inicial)= do, numero, identificador
P(Exp)= numero, identificador, λ
P(Factor)= numero, identificador

c)
P(Inicial)= do, numero, identificador
P(Exp)= numero, identificador, λ
P(Factor)= numero, identificador, λ

d)
P(Inicial)= do, numero, identificador, λ
P(Exp)= numero, identificador, λ
P(Factor)= numero, identificador, λ

e) Ninguna de las anteriores.

19. – Respecto a las siguientes producciones:
T → F T1
T1 → T2 F T1
T1 → λ
T2 → *
T2 → /
¿Cuántas veces tendremos que añadir los siguientes del no terminal que está a la izquierda de la producción para calcular los símbolos siguientes de todos los símbolos no terminales que aparecen en las producciones (considerar una única iteración del algoritmo)?

a) 2
b) 3
c) 1
d) Ninguna
e) Ninguna de las anteriores.

20.- Dada la siguiente gramática de expresiones, queremos añadir la declaración de variables con la posibilidad de realizar tanto una inicialización de variable con tipo de datos, como automática. Ejemplo: Var a: integer; Var b:integer = 10; Var c = 10; ¿Cuál sería la ampliación correcta de la gramática?
Rule <tipus>::=
Integer | Real | Character | Bool | ^ <tipus> |
Array "[" numero "]" of <tipus> | identificador
Rule <DecVar>::=Var identificador : <tipus> ";"
Rule <ExpArit>::=<Terme>{("+"|"-")<Terme>}
Rule <Terme>::<Factor>{("*"|"/")<Factor>}
Rule <Factor>::numero|identificador ["="<ExpArit>]

a) <DecVar>::= Var identificador ((:<tipus> ["=" <ExpArit>]) | "=" <ExpArit>) ";"
b) <DecVar>::= Var identificador [:<tipus>] ["=" <ExpArit>] ";"
c) <DecVar>::= Var identificador ([:<tipus>] ["=" <ExpArit>] | ["=" <ExpArit>]) ";"
d) <DecVar>::= Var identificador (:<tipus> "=" <ExpArit> | "=" <ExpArit>]) ";"
e) Ninguna de las anteriores.

ANÁLISIS SEMÁNTICO

21.- ¿Dónde se guarda el significado de los símbolos terminales?
a) En la tabla de símbolos.
b) En los parámetros de las reglas BNF.
c) En las variables locales de las reglas BNF.
d) b) y c).
e) Ninguna de las anteriores.

22.- ¿Qué se ha de añadir a una gramática para implementar el análisis semántico?
a) Símbolos de acción.
b) Símbolos terminales.
c) Símbolos no terminales.
d) Producciones.
e) Ninguna de las anteriores.

23.- ¿Cómo se representa el tipo de datos array de n dimensiones?
a) Con una estructura TArray con la lista de las n dimensiones y el tipo de los elementos del array.
b) Recursivamente con n estructuras TArray anidadas.
c) Con un espacio de memoria cuyo tamaño es el producto de las n dimensiones por el tamaño de los elementos del array.
d) Con una lista encadenada de los elementos del array.
e) Ninguna de las anteriores.

24.- ¿Cuáles son las principales acciones que realiza el análisis semántico de declaraciones?
a) Verificar que no haya declaraciones duplicadas.
b) Crear una representación del significado que se asocia a los símbolos que aparecen en la declaración.
c) Añadir las entradas a la tabla de símbolos correspondientes a lo que se declara.
d) Todas las anteriores.
e) Ninguna de las anteriores.

25.- ¿Cuáles son las principales acciones que realiza el análisis semántico de expresiones?
a) Verifica que se hayan declarado los símbolos que aparecen en la expresión.
b) Comprueba que las operaciones son compatibles con los tipos de datos de sus operandos.
c) Calcula el tipo de datos del resultado de las operaciones.
d) Todas las anteriores.
e) Ninguna de las anteriores.

26.- ¿Qué acciones semánticas se han de hacer para seleccionar la sobrecarga concreta de un método que se ha de aplicar en una llamada?
a) Se ha de comparar la lista de tipos de los parámetros de la llamada con la lista de tipos de los parámetros de cada una de las posibles sobrecargas. Se selecciona la sobrecarga que coincidan los tipos de datos de sus parámetros con los de la llamada.
b) a) y en caso de no darse ninguna coincidencia, se aplicarán las coerciones de clase derivada a clase base a los parámetros de la llamada hasta que coinciden con los tipos de datos de los parámetros de una de las sobrecargas, que será la seleccionada. Si no hay coincidencia se genera el error de que no se puede hacer la llamada.
c) a) y en caso de no darse ninguna coincidencia, se aplicarán las coerciones de clase derivada a clase base a los parámetros de la llamada hasta que coinciden con los tipos de datos de los parámetros de una o más sobrecargas. Si sólo hay coincidencia con una sobrecarga, esta serla la seleccionada. En el caso que haya más de una se genera el error de llamada ambigua y en caso de ninguna se genera el error de que no se puede hacer la llamada.
d) b) y se verifica que el tipo de datos del resultado de la llamada se pueda construir por copia.
e) Ninguna de las anteriores.

27.- ¿Qué comprobaciones semánticas se tendrían que hacer en la regla BNF siguiente?

```
Rule <Instruccio>::=
    .. |
    While <Expressio> do <instruccio> |
    ..
```

a) Que el tipo de datos que devuelve el descriptor de valor asociado a Expressio sea TInt
b) Que el tipo de datos que devuelve el descriptor de valor asociado a Expressio sea TBool
c) a) y que el campo LValue del descriptor de valor asociado a Expressio sea true para asegurar que no estamos en bucle infinito
d) b) y que el campo LValue del descriptor de valor asociado a Expressio sea true para asegurar que no estamos en bucle infinito
e) Ninguna de las anteriores.

28.- ¿Cuántas entradas de cada tipo contiene la pila de la tabla de símbolos de LOOS impresa en el punto indicado?

```
Class a  {
        m_c1:Integer;
        Procedure p1(&x:Integer,y:Real);
}
Procedure a::p1(&x:Integer,y:Real) {
        var b:array [5]of array [10] of integer;
}
Procedure main() {
        var a:integer;
        var b: real;
        IMPRIMIR_TAULA_DE_SIMBOLS;
}
```

a) ETSTipus(1), ETSFuncio (2), ETSVariable (2), TSeparador (1), TSeparadorFuncio (1)
b) ETSTipus(1), ETSFuncio (1), ETSVariable (2), TSeparador (1), TSeparadorFuncio (1)
c) ETSClasse(1), ETSFuncio (1), ETSVariable (2), TSeparador (1), TSeparadorFuncio (1)
d) ETSClasse(1), ETSFuncio (2), ETSVariable (2), TSeparador (1), TSeparadorFuncio (1)
e) Ninguna de las anteriores.

29.- ¿Cuantas comprobaciones relacionados con tipos de datos (comparaciones y cálculos de tipos) hace el compilador de LOOS en la expresión ((20+5)*3.0) / 4.0?

a) 6
b) 5
c) 8
d) 7
e) Ninguna de las anteriores.

30.- ¿Qué comprobaciones semánticas ha de hacer LOOS en la regla <ElementClasse> en la parte de la creación de campos de los objetos?

```
Rule <ElementClasse(classe:TClasse,nomRec)>::=
    // Camp ------------------------------------------------------------
    @var nom,t;
    identificador#(nom) R1
    : <tipus(t,nomRec)> ";" R2
    |…
```

a) **R1:** @TS.ComprovarDuplicat(nom);
R2: @TS.Insertar(ETSVariable(nom,t,unbound));
b) **R1:** @classe.ComprovarDuplicat(nom);
R2: @TS.Insertar(ETSClassField(nom,t,unbound));
c) **R1:** @classe.ComprovarDuplicat(nom);
R2: @classe.Insertar(ETSVariable(nom,t,unbound));
d) **R1:** @classe.ComprovarDuplicat(nom);
R2: @classe.Insertar(ETSClassField (nomRec,nom,t,unbound));
e) Ninguna de las anteriores.