3er Guía Compiladores

1. Una gramática no es ambigua si existe alguna cadena de terminales que pueda obtenerse mediante dos árboles de análisis sintáctico distintos	F
2. Dos gramáticas son equivalentes si generan el mismo lenguaje	v
3. En el análisis sintáctico ascendente el árbol de análisis sintáctico la construcción se inicia en la raíz y avanza hacia las hojas	F
4. En el análisis sintáctico descendente se construye el árbol de análisis sintáctico de la cadena desde las hojas y avanza hacia la raíz	F
5. El árbol sintáctico no es una versión condensada del árbol de análisis sintáctico	F
6. Un esquema de traducción asocia a cada elemento de una GLC un conjunto de atributos y a cada producción, un conjunto de reglas semánticas para calcular los valores de los atributos asociados con los símbolos que aparecen en esa producción	F
7. Definición dirigida por la sintaxis es una GLC en la que se encuentran intercalados, en los lados derechos de las producciones, fragmentos de programa llamados acciones semánticas	F
8. Los valores de los atributos sintetizados se calculan a partir de los valores de atributos de su nodo padre o sus nodos hermanos	F
9. Un atributo es heredado si su valor depende de los atributos de su padre y/o de sus hermanos	V
10. El código de tres direcciones consiste en una secuencia de instrucciones, cada una de las cuales tiene máximo tres operandos	V
11. En lenguaje C los parámetros formales son como variables locales que ya fueron inicializadas en el momento de llamada a la función o procedimiento	V
12. En el lenguaje C las variables locales (no estáticas) se crean cuando se entra a una función y se destruyen cuando se sale de la función	V
13. En hoc los parámetros formales no tienen nombre	V
14. No es posible definir funciones recursivas en hoc	F
15. En hoc no hay variables locales	V
16. En hoc cuando una función termina su ejecución se saca su marco de la pila de llamadas	V
17. En hoc los parámetros reales son listas de expresiones	V
18. En hoc el código que ejecuta la máquina virtual de pila está en prefijo	F

1. Indica gráficamente cómo del símbolo inicial de una gramática deriva una cadena del lenguaje
    a. Árbol de análisis sintáctico con anotaciones
    b. Árbol de análisis sintáctico
    c. Árbol sintáctico
    d. Ninguno de los anteriores

2. Un código de tres direcciones se usa en
    a. El análisis sintáctico
    b. Generación de código intermedio
    c. Análisis léxico
    d. Generación de código

3. Un _____ es [A->α.β,α] donde A->α.β es una producción y a es un terminal o $
    a. mango
    b. prefijo variable
    c. elemento LR(1)
    d. elemento LR(0)

4. Es una producción de G con un punto en cierta posición del lado derecho
    a. mango
    b. prefijo variable
    c. elemento LR(1)
    d. elemento LR(0)
5. Son prefijos de las formas de frase derecha que pueden aparecer en la pila
    a. mango
    b. prefijo viable
    c. elemento LR(1)
    d. elemento LR(0)

6. Un _____ de una forma de frase derecha g es una producción S->α y una posición de g donde la cadena α podría encontrarse y sustituirse por A para producir la forma de frase derecha previa en una derivación por la derecha de g
    a. mango
    b. prefijo variable
    c. elemento LR(1)
    d. elemento LR(0)

1. Construir explícita o implícitamente el grafo de dependencias
2. Construir el a.a.s para la gramática y las entradas dadas
3. Evaluar las reglas semánticas de acuerdo con el orden topológico
4. Supuesto que el grafo de dependencias determina un orden parcial construir un orden topológico compatible con el orden parcial

7. Para la realización de una traducción dirigida por la sintaxis el orden sería
    a. 1,2,3,4

b. 1,3,4,2
c. 2,1,4,3
d. 4,3,2,1

1. Ejecutar la función (poner el contador de programa igual a la dirección de su primera instrucción y ejecutar la instrucción a la que apunta el contador del programa) y meter el valor de retorno de la función en la pila
2. Meter los parámetros en la pila y meter el marco de la función en la pila de llamadas
3. Poner el contador de programa igual a la dirección de retorno y ejecutar la instrucción a la que apunta el contador de programa
4. Sacar parámetros de la pila y sacar marco de la pila de llamadas

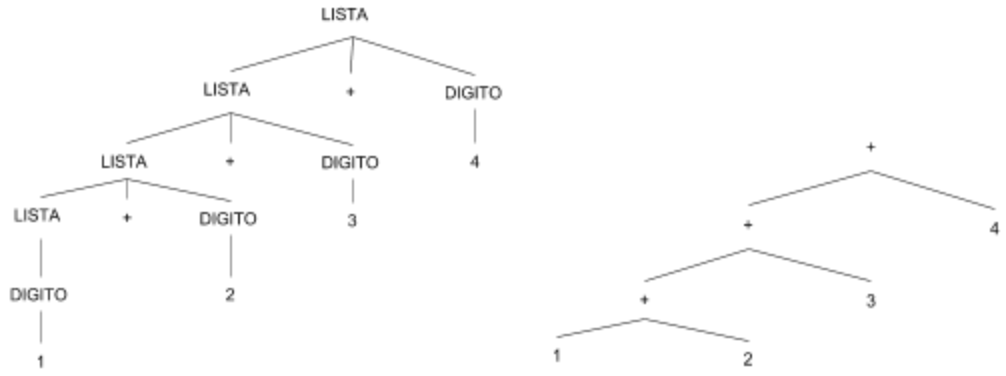
8. ¿De acuerdo al mecanismo de llamada de función cuál es el orden correcto?
    a. 1,2,3,4
    b. 1,3,4,2
    c. 2,1,4,3
    d. 4,3,2,1

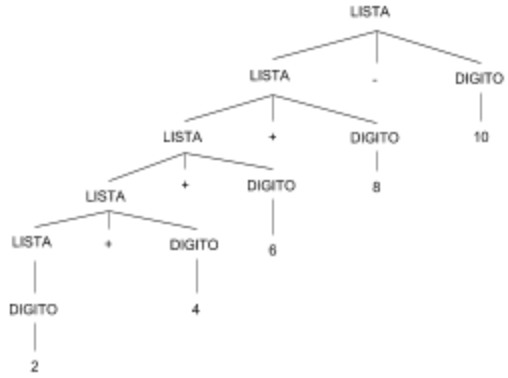
PROBLEMAS

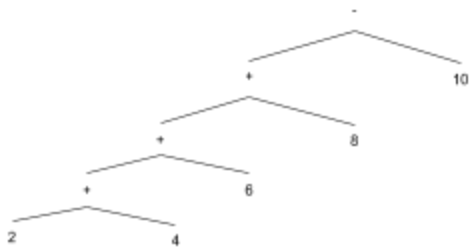
Dada la gramática

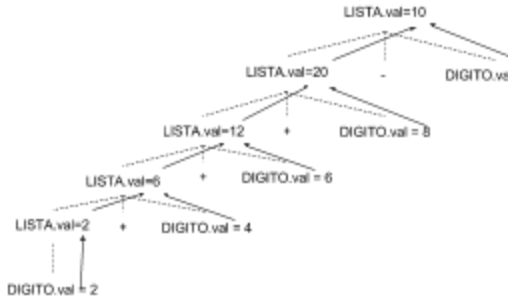
LISTA -> LISTA + DIGITO | LISTA – DIGITO| DIGITO
DIGITO -> 0|1|2|3|4|5|6|7|8|9

1. Dibuje el árbol de análisis sintáctico, el árbol sintáctico y el grafo de dependencias para las siguientes cadenas:
   1+2+3+4 y 2+4-6+8-10

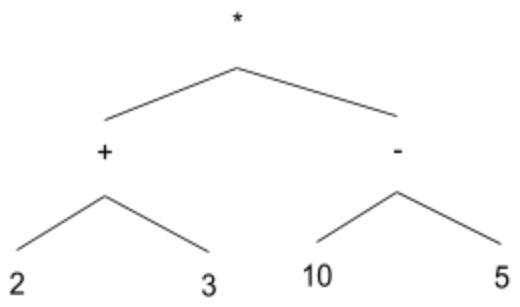


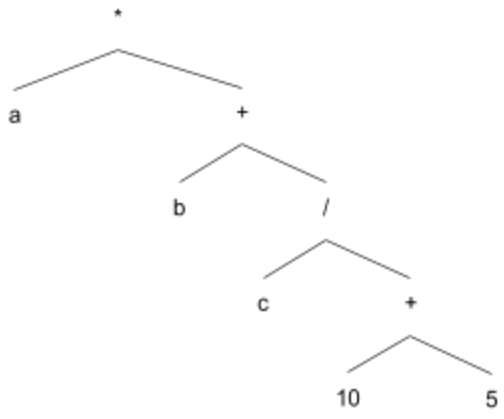






2. Dibuje el árbol sintáctico para las siguientes cadenas
   (2+3)*(10-5) y a/(b+c/(d+e))





3. Escriba las expresiones de tipo para:
   a. Un entero

      integer

b. Un apuntador a entero
   pointer(integer)
c. Un arreglo de apuntadores a char
   array(1..n,pointer(char))
d. struct agregado {char x;int y;double z;};
   record(agregadoXrecord((xXchar)x(yXint)x(zXdouble)))

4. Supóngase que los nombres de tipos enlace y nodo se definen como en la sección 6.3 del libro del dragón. ¿Cuáles de las siguientes expresiones de tipos son estructuralmente equivalentes?
   enlace
   pointer nodo
   pointer enlace
   pointer(record((infoXinteger)X(siguienteXpointer(nodo))))

5. Escriba el código de 3 direcciones de (a * b + h) – j * k + 1

```
t1:=a*b;
t2:=t1+h;
t3:=j*k;
t4:=t2-t3;
t5:=t4+1;
```

6. Hacer el código 3 direcciones de la expresión: a>b + h or b == d

```
t1:=b+h
t2:=a>b
t3:=b==d
t4:=t2 or t3
```

7. Hacer el código de 3 direcciones de:

a:= 2*x+10;
while(a<=p+2) {
a:=p[4+i*2];
}
a:=a+1
	for(i=0;i<5;i++){
	A;
	}

t1:=2*x; t2:=t1+10	t1:= 0 L1: if t1>=5 goto L2;

a:=t2;
t3:=p+2;
 	
L1: t4 = *t3;
if a>t4 goto L2;
t5:=i*2;
t6=4+t5;
t7:=p+t6;
a:=*t7;
L2: a:=a+1;
	A;
	t1:= t1+1;
	L2:

8. Si una instrucción de asignación de la forma x=y+z. Se traduce a:	Cómo se traducen

mov y, R0
add Z, R0
mov R0, x
	a= b + c;
	d = a + c;
	a = a +1 ;

mov b, R0
add c, R0
mov R0, a
mov a, R0
add c, R0
mov R0, d
mov a, R0
add 1, R0
mov R0, a