Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

TRADUCTORES, COMPILADORES E INTÉRPRETES. EJERCICIOS YACC. TEMA 4.

1.) (AHO, 5.4) Dar una definición dirigida por sintaxis, y el programa LEX/YACC correspondiente, para traducir expresiones infijas (con las operaciones + y * solamente), en expresiones infijas en las que se han eliminado los paréntesis redundantes. Por ejemplo, dado que + y * se asocian a la izquierda, se tiene que
((a*(b+c))*(d))
se debe traducir a
a*(b+c)*d.

2.) (AHO 5.5) Dar una definición dirigida por sintaxis, y el programa LEX/YACC correspondiente, para obtener la derivada de expresiones formadas mediante los operadores aritméticos + y * a la variable **x**, y a las constantes. No es necesario efectuar ninguna simplificación. P.ej., la expresión

3*x

se debe traducir a

0*x+3*1.

3.) (AHO 5.6) La siguiente gramática genera expresiones formadas por constantes enteras y reales a las que se aplica el operador aritmético de la suma +. Cuando se suman dos enteros el resultado es entero, y si no, es real.

E → E + T | T
T → **num . num** | **num**

Hacer un programa LEX/YACC que permita
a) Determinar el tipo de cada subexpresión.
b) Además de lo anterior, que permita traducir expresiones a notación posfija. Puede usarse el operador unario **int_to_real** para convertir un valor entero en su equivalente real, de manera que los dos operandos del + en la forma posfija deben ser del mismo tipo.

4.) En la siguiente gramática, una asignación también se considera una expresión, con el mismo significado que en C.

S → E
E → E := E | E + E | (E) | **id**

P.ej., la expresión *b := c*, asigna a *b* el valor de *c*, y lo que es más, la expresión *a:=(b:=c)* asigna a *b* el valor de *c*, y a *a*, el valor de *c* también.

Construir un programa LEX/YACC para chequear que la parte izquierda de una asignación es un l-valor, o lo que en nuestro caso es lo mismo, un identificador.

5.) Hacer un programa LEX/YACC que permita simular la declaración de variables y su posterior uso, de forma que se detecten las variables redeclaradas y las que se usan sin haberse declarado. Así, las siguientes entradas darían los mensajes indicados:

DECLARAR uno, dos;
USAR dos, tres;
> *tres no declarado.*
DECLARAR uno, tres, cuatro;
> *uno ya declarado.*

6.) Sea la siguiente gramática para declarar variables

D → **id** L
L → **,** **id** L | **:** T
T → **integer** | **real**

Construir un programa LEX/YACC que lea una declaración, cree una lista con los identificadores declarados; junto con cada identificador se deberá guardar el tipo de éste. Al final del análisis se visualizará la lista con todos los identificadores y el tipo de cada uno.

7.) Construir un programa LEX/YACC que acepte declaraciones de funciones de la forma:

DECLARAR Nombre_función(lista_parámteros_formales);

y que permita usar dichas funciones de la forma:

USAR Nombre_función(lista_parámetros_reales);

El analizador debe controlar lo siguiente:

- Que no hay redeclaraciones de funciones.
- Que los parámetros formales de la función son sólo variables.
- Que una variable no posee el mismo nombre que una función.
- Que al usar una función, ésta ha sido previamente declarada.
- Que el número de parámteros reales al usar una función coincide con el número de parámteros formales indicados en su declaración.

No se tendrá en cuenta tipo alguno para las variables ni para las funciones. Además, como parámetro real de una función se permiten llamadas a función también.

8.) (AHO 5.13) Reescribir la gramática presentada en el ejercicio 4.), de manera que agrupe las asignaciones a la derecha, y las sumas a la izquierda, y dé mayor prioridad a la suma que a la asignación. No hacerlo con YACC, sino reescribiendo la gramática.

9.) (AHO 5.10) Sea un traductor en el que podemos colocar etiquetas entre el código generado, etiquetas a las que se pueden producir saltos mediantes instrucciones a tal efecto; ej.:

```
        jmp etiq2
        :
        código
etiq2:
        :
        código
```

Supongamos que el código de las operaciones e instrucciones puede ser generado a través de la función **genera_código()**, excepto el referente a las instrucciones de salto que puede producir la instrucción **break** dentro de un bucle **while**. Ej.:

```
while expr1 do begin                                    código del while1
        S11                                             cód. de S11
        break                                           jmp etiq1
        S13                                             cód. de S13
        while expr2 do begin        se traduce en       código del while2
                S141                                    cód. de S141
                break                                   jmp etiq2
        end;                                    etiq2:
        S15                                             cód. de S15
end;                                            etiq1:
S2                                                      cód. de S2
```

Suponer que se tiene la función **genera_etiqueta()** que cada vez que se la llama da un nombre de etiqueta diferente. Construir el esquema de traducción que consiga generar el código, los saltos y las etiquetas, tal y como se ve en el ejemplo propuesto. Asimismo, dibujar el grafo

de dependencias obtenido en dicho ejemplo.

10.) (AHO 5.21) Sea la siguiente gramática

L → M L **b** | **a**

M → **ε**

Dibujar el árbol sintáctico que reconoce la sentencia *abbb*. Justificar verbalmente por qué esta gramática no es LR(1).

11.) Hacer un intérprete utilizando LEX y YACC que permita manipular cadenas de caracteres, y que permita la suma y la resta. La suma de dos cadenas, producirá otra cadena cuyo resultado será la concatenación de las primeras. La resta se producirá entre una cadena y un entero, de la forma **c - n**, y dará como resultado la cadena **c** pero sin los **n** últimos caracteres; si la cadena **c** tiene más de **n** caracteres, se producirá un error; si el número **n** es negativo, en lugar de quitarse de **c** los **n** último caracteres, se quitarán los |**n**| primeros caracteres. Las cadenas se encierran entre comillas dobles y los números serán enteros incluido el cero, y sin decimales.