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

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria extraordinaria de Febrero de 2002
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____ ***Calificación:***

TEORÍA

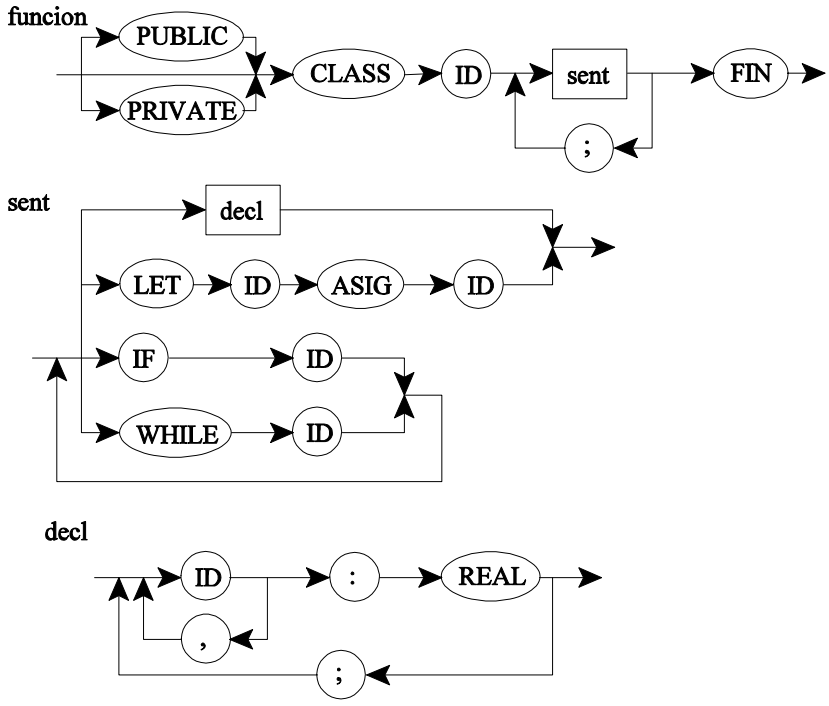
1.- Partiendo del esquema de traducción visto en clase para generar código de tercetos, supongamos que lo simplificamos y nos quedamos sólo con aquella parte del lenguaje que reconoce sentencias de asignación y genera el código de tercetos correspondiente.
a) ¿Cuál sería el código generado para la sentencia?

```
x = (a + b) * (c + d * f) * (e + a)
```

b) Proponer el funcionamiento de un descompilador de dicho sublenguaje, esto es, un sistema que recoja el código de tercetos anterior y regenere la sentencia de asignación original.

2.- Responder a las siguientes cuestiones:
a) ¿Qué ocurre si, mientras se realiza un análisis LALR a través de sutabla correspondiente, nos encontramos con que es necesario reducir por una regla cuyo consecuente no se encuentra encima de la pila de trabajo?
b) Una tabla de análisis LL(1) ¿puede contener más de una regla en cada casilla?
c) Una gramática con ε, ¿impide un análisis descendente con retroceso?
d) Una gramática recursiva por la derecha, ¿es imposible de ser gestionada por algún análisis con retroceso?

3.- Dados los siguientes diagramas de Conway:



crear las funciones recursivas expresadas en pseudocódigo que a partir de tales diagramas implementen un análisis descendente con recursión y sin retroceso.

PRÁCTICA

Se pretende diseñar un sistema de generación de código para visualizar los valores de una función de dos variables en un rango cuadrado determinado.

Para ello, se introduce un nuevo terceto de la forma:

write *variable*

que visualiza el valor de *variable* por la salida estándar.

Las entradas tienen la forma:

```
FUNCTION (X, Y)= X+Y*7.8-(Y*Y)
X UPPER LIMIT IS 10.5
X LOWER LIMIT IS 5.5
X STEP IS 0.5
Y UPPER LIMIT IS 20.0
Y LOWER LIMIT IS 15.0
Y STEP IS 0.25;
```

donde:

- El objetivo es transformar un bloque como éste en un bloque de código de tercetos formado por dos bucles anidados que recorren la variable X y la variable Y, y para cada par (X, Y) de su rango, se visualiza el valor de la función dada. En el caso anterior, el código generado es:

```
     Y=15.000000
label etq1
     if Y>20.000000 goto etq3
     X=5.500000
label etq2
     if X>10.500000 goto etq4
     tmp1=7.800000
     tmp2=Y*tmp1
     tmp3=X+tmp2
     tmp4=Y*Y
     tmp5=tmp3-tmp4
     write tmp5
     X=X+0.500000
     goto etq2
label etq4
     Y=Y+0.250000
     goto etq1
label etq3
```

- La primera línea contiene la descripción de la función a evaluar. Puede usarse cualquier operador aritmético, así como el menos unario y los paréntesis.
- Cualquier constante debe ser de tipo real, esto es, contiene un punto decimal y al menos un dígito antes y otro después de dicho punto.
- La única línea obligatoria es la primera.
- Las líneas que definen los límites y el paso pueden estar en cualquier orden, y un mismo límite o paso puede repetirse, considerando que la última aparición es la válida (ver gramática).
- Si un límite o paso no aparece, se le dará un valor por defecto, teniendo en cuenta que el valor por defecto para un límite inferior es 0.0, para uno superior es 10.0, y para un paso es 1.0.
- Si, una vez consumida la entrada de una función, el límite superior de una variable es menor que su límite inferior debe emitirse un mensaje de error.
- Se permiten comentarios del estilo C, esto es englobados entre /* y */.
- Se permite la especificación de varias funciones con sus límites y pasos correspondientes, separadas por comas.

El esqueleto Lex/Yacc es el siguiente:

Fichero Exfeb02l.lex

Fichero Exfeb02y.yac

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

%}

%union{
	float numero;
	struct {
		char texto[200];
		char temporal[10];
	} expresion;
}

%%
funciones	:	/* Epsilon */
		|	funciones funcion ';'
		|	funciones error ';'		{		}
		;
```

```
funcion         :       FUNCTION '(' X ',' Y ')' '=' expr limites       {
                        }
                ;
limites :       /* Epsilon */
                |       limites X UPPER LIMIT IS REAL   {                       }
                |       limites X LOWER LIMIT IS REAL   {                       }
                |       limites X STEP IS REAL          {                       }
                |       limites Y UPPER LIMIT IS REAL   {                       }
                |       limites Y LOWER LIMIT IS REAL   {                       }
                |       limites Y STEP IS REAL          {                       }
                ;
expr            :       REAL    {

                        }
                |       X       {

                        }
                |       Y       {

                        }
                |       expr '+' expr   {
```

```
                        }
                |       expr '*' expr    {

                        }
                |       expr '/' expr    {

                        }
                |       expr '-' expr    {

                        }
                |       '-' expr %prec MINUS_UNARIO      {

                        }
                |       '(' expr ')'        {

                        }
                ;
%%

void main(){
        yyparse();
}
void yyerror(char *s) {
        printf("%s\n", s);
}
/* Otras funciones */
```

Se pide: rellenar el esqueleto Lex/Yacc dado con objeto de conseguir las funcionalidades pedidas.