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

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria ordinaria de Septiembre de 2003
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____

Calificación: _____

TEORÍA

1.- Escribir una gramática en Yacc que sea capaz de reconocer las líneas de un glosario, esto es, cada línea comienza con una palabra y viene seguida por una lista de números naturales separados por comas. Se asume que una palabra es cualquier secuencia de caracteres diferentes de los dígitos. Los espacios entre los números naturales se ignoran. Escribir asimismo, usando notación Lex, las expresiones regulares encargadas de reconocer los *tokens*. Por ejemplo, las siguientes líneas deben ser reconocidas:
ontología 23, 45, 67
semántica 102, 107, 201
estado externo 208, 209, 206

2.- Sea la siguiente gramática:

```
?   view  :   VIEW expr
          ;
?   expr  :   ID '(' VALOR ')'
?         |   expr AND expr
?         |   expr OR expr
          ;
```

y su tabla de análisis LALR(1) asociada:

	VIEW	ID	VALOR	AND	OR	(	)	$	view	expr
0	D2								1	
1								Aceptar		
2		D4								3
3	R1	R1	R1	D5	D6	R1	R1	R1		
4						D7				
5		D4								8
6		D4								9
7			D10							
8	R3	R3	R3	R3	R3	R3	R3	R3		
9	R4	R4	R4	D5	R4	R4	R4	R4		

10							D11			
11	R2	R2	R2	R2	R2	R2	R2	R2		

Indicar el árbol sintáctico que reconoce la cadena:

VIEW ID (VALOR) AND ID (VALOR)

anotando en el árbol en qué orden se aplican las reglas. Como resumen, decir cuántas reducciones se hacen, y cuántos desplazamientos hasta llegar a la aceptación final.

3.- Dado el siguiente texto de entradaal traductor que generad código de tercetos visto en clase, indicar exactamente el código de tercetos que se generaría:

```
IF a>6 THEN
        IF B=0 THEN
                a:=6;
        ELSE
                a:=-a;
        FIN IF;
FIN IF;
```

Recuérdese que el único lugar en el que se admite la aparición de una constante de tipo numérico es en una asignación simple (también llamada copia).

4.- Definir los siguientes conceptos:

- Compilador cruzado.
- Metacompilador.
- *Linkador* (enlazador).
- Recolección de basura.

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

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria ordinaria de Septiembre de 2003
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____

Calificación: _____

PRÁCTICA

Varios amigos desean ser coautores de un libro, y cada uno de ellos escribe por separado un capítulo en el procesador de textos Págica. A la hora de pegar los capítulos se dan cuenta de que el procesador es muy antiguo y no lo permite, por lo que la tabla de contenidos y el índice analítico deben generarse de manera independiente para, posteriormente, unirlos a mano.

Este ejercicio se va a centrar únicamente en la unión de los índices aalíticos independientes generados por cada coautor en su capítulo. Cada línea del índice analítico tiene la forma del ejercicio 1 del apartado de teoría, esto es, la línea comienza con un texto (también llamado **término**) y para cada término se indican a continuación , y separados por comas todas y cada una de las páginas en las que es referenciado.

De esta manera, cada coautor genera su índice analítico que consta de todas las palabras clave del libro que se desean registrar junto con los números de página de su capítulo en que aparecen. Pues bien, el objetivo del presente ejercicio consiste en construir los programas Lex/Yacc que permitan fusionar correctamente los distintos ficheros en uno solo, que tenga una sola línea por cada palabra diferente. Para facilitar esta labor supondremos que todos los subíndices analíticos se yuxtaponen en un solo. Por ejemplo, si hay tres coautores y el índice analítico consta de cuatro palabras: electrón, protón, neutrón, taquión, cada uno de ellos generaría un fichero distinto:

coautor 1:

electrón 12, 34, 56
protón 23, 67
neutrón
taquión 12, 34, 60

coautor 2:

electrón 75, 90
protón 85, 90, 101
neutrón
taquión

coautor 3:

electrón
protón 115, 123, 156
neutrón
taquión 125, 201, 202, 203

pero supondremos que a nuestro programa le entra la yuxtaposición de todos ellos (que puede obtenerse con un fichero batch de MS-DOS, p.ej.), o sea:

electrón 12, 34, 56

protón 23, 67
neutrón
taquión 12, 34, 60
electrón 75, 90
protón 85, 90, 101
neutrón
taquión
electrón
protón 115, 123, 156
neutrón
taquión 125, 201, 202, 203

De esta forma, la salida de nuestro programa debe ser la adecuada fusión de todas las líneas, teniendo en cuenta que las palabras que no aparecen en ninguna página en ninguno de los capítulos no deben aparecer en el resultado final. En nuestro caso se obtendría:

electrón 12, 34, 56, 75, 90
protón 23, 67, 85, 90, 101, 115, 123, 156
taquión 12, 34, 60, 125, 201, 202, 203

Sea la siguiente gramática Yacc que reconoce el fichero de entrada:

```
%{
// Declaraciones e inclusiones necesarias (apartado 1).
%}
%union{
// Declaración de atributos (apartado 2).
}
// Declaración de tokens y asignación de atributos (apartado 3).
%%
texto                   :     texto linea
                        |     /* Epsilon */
                        ;
linea                   :     PAL    {    Acción    1ª    (apartado    4). }
                              listaNumerosOpcional
                        ;
listaNumerosOpcional    :     listaNumeros
                        |     /* Epsilon */
                        ;
listaNumeros            :     numero
                        |     listaNumeros ',' numero
                        ;
numero                  :     NUM { Acción 2ª (apartado 5). }
                        ;
%%

// Otros includes y funciones adicionales necesarias (apartado 6).
```

Se pide:

- Rellenar los 6 apartados que se proponen en el esqueleto anterior. Escribir las soluciones dejando bien claro a qué apartado corresponde cada solución aportada.

- Como **apartado 7** escribir el programa Lex asociado al programa Yacc que se propone.