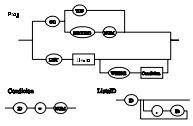
Examen de Traductores, Intérpretes y Compiladores.
Convocatoria ordinaria de Junio de 1999.
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre:
_____

Calificación: _____

TEORÍA



1.- Sean los siguientes diagramas de Conway:
Crear las funciones recursivas, expresadas en pseudocódigo, que a partir de estos diagramas implementan un análisis descendente con recursión y sin retroceso.

2.- Qué problema/s posee la siguiente gramática?
Si dicha gramática se usa en un reconocimiento LR(1), posee algún otro problema?

```
S    :    a b S
     |    a b
     |
;
```

Escribir una gramática equivalente que no posea tales defectos.

3.- Responder a las siguientes cuestiones:

a) El análisis descendente con retroceso, permite reconocer gramáticas recursivas a izquierda? Y el ascendente con retroceso? Justificar la respuesta.

b) Cuáles son las reglas que sigue Lex en el reconocimiento de patrones?

c) Cuáles son los métodos básicos de recuperación de errores sintácticos?

4.- Sea la siguiente gramática y su tabla LALR(1):

```
        s       :   ID ASIG e
                |   ID ASIG s
        ;
        e       :   e '+' e
                |   e '*' e
                |   '(' e ')'
                |   ID
        ;
```

Estado	ID	ASIG	0	0	(	)	$	e	s
0	D2								1
1							Acep.		
2		D3							
3	D7				D6			4	5
4	R1	R1	D8	D9	R1	R1	R1		
5	R2	R2	R2	R2	R2	R2	R2		
6	D11				D6			10	
7	R6	D3	R6	R6	R6	R6	R6		
8	D11				D6			12	
9	D11				D6			13	
10			D8	D9		D14			
11	R6	R6	R6	R6	R6	R6	R6		
12	R3	R3	R3	D9	R3	R3	R3		
13	R4	R4	R4	R4	R4	R4	R4		
14	R5	R5	R5	R5	R5	R5	R5		

Reconocer o rechazar la cadena

ID ASIG ID ASIG '(' ID ')' ID '*' ID

Examen de Traductores, Intérpretes y Compiladores.
Convocatoria ordinaria de Junio de 1999.
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre:
_____

Calificación: _____

PRÁCTICA

Por motivos históricos, la empresa en que trabajamos almacena los datos de sus empleados en un simple fichero de texto donde cada línea es un registro destinado a un empleado, y cada línea almacena los siguientes datos:

Nombre, Apellidos, DNI, Sueldo, n hijos

por ese orden.

Hasta ahora nuestra empresa ha trabajado con programas desarrollados en C, y que hacían accesos a dicho fichero de texto mediante un módulo de programa llamado Fichero.c, y que contiene las siguientes operaciones básicas sobre el fichero de datos de empleados en formato texto:

void abrir(). Abre el fichero de datos y se posiciona sobre el primer registro si lo hay.

void cerrar(). Cierra el fichero de datos.

int siguiente(). Pasa al siguiente registro del fichero de datos. Devuelve 1 si ya no hay más, y 0 en caso contrario.

char * campo(int n). Se le pasa un número del 1 al 5, que identifica la posición del campo cuyo valor se desea saber. 1-Nombre, 2-Apellidos, 3-DNI, etc. Devuelve la cadena con el valor del dato pedido. Todos los datos son interpretados como cadenas de texto.

int mieof(). Devuelve 1 si ya no hay más registros, y 0 en caso contrario.

Nota: Como puede verse, tratamos sólo de los accesos para lectura, y no nos interesa la escritura o modificación del fichero de datos.

Pero este ha sido el método de funcionamiento hasta ahora. Con nuestra llegada a la empresa, la dirección ha decidido flexibilizar el acceso a este fichero, y para ello desea que hagamos un peque o intérprete que permita a cualquier usuario acceder al fichero de texto de una forma mucho más flexible. La gramática que se le ha ocurrido a dirección es la siguiente:

```
prog    :       prog sent ';'
        |       prog error ';'
        |
        ;
sent    :       GO TOP
        |       GO RECORD NUMERO
        |       LIST lista_campos WHERE condicion
        |       LIST lista_campos
```

```
        ;
lista_campos  :     campo
              |     lista_campos ',' campo
              ;
campo :       NOMBRE
        |     APELLIDOS
        |     NIF
        |     SUELDO
        |     HIJOS
        ;
condicion     :     campo '=' TEXTO
              ;
```

Como puede verse, este intérprete permite que se le introduzcan sentencias y debe efectuar una acción inmediatamente. Las sentencias que se permiten y lo que deben hacer son:

GO TOP: Se coloca en el primer registro del fichero.

GO RECORD n: Se coloca en el **n**-ésimo registro del fichero.

Sólo se permiten valores mayores de 0.

El valor de **n** no puede ser superior al número de registros del fichero.

LIST campo1, campo2, ...: Lista el valor de los campos especificados para todos los registros desde aquél en que estamos colocados hasta el final.

Cada registro se visualiza en una línea independiente.

El usuario debe elegir al menos un campo a visualizar.

*LIST campo1, campo2, ... WHERE campo**n** = "valor**n**"*: Igual que la anterior, pero sólo visualiza aquellos registros cuyo valor del campo**n** coincide con la cadena de texto valor**n**.

La operación campo() de Fichero.c retorna los valores sin comillas.

Por tanto, por cada sentencia de las anteriores que el usuario introduzca en nuestro intérprete, dicho intérprete debe transformarla en la correspondiente secuencia de llamadas a las funciones que hay en Fichero.c. P.ej., en el caso de LIST deberemos hacer llamadas a siguiente() y visualizar los datos de los registros por los que vayamos pasando.

Se pide:

Realizar los programas Lex y Yacc completos que permitan el intérprete que se describe, y que se sigue la gramática anterior.