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 2003
3er Curso de I.T. Informática de Sistemas.

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

TEORÍA

1.- Dada la gramática que reconoce la construcción un tipo complejo, y la declaración de variables de tales tipos (tal y como se ha visto en clase: **POINTER TO**, **ARRAY OF**, etc.), hacer **un único** diagrama de Conway equivalente, así como la función asociada a dicho diagrama que reconoce el lenguaje que ésta acepta.

Nota: No contemplar la utilización de variables.

Nota: La función sólo debe contemplar aspectos sintácticos, no semánticos.

2.- Sea el conjunto de *tokens* T={a, b, c}. Construir una gramática G=(N, T, P, S) que reconozca cualquier cadena de palíndromos (palabras capicúas) formada por elementos de T. La gramática construida ¿es LL(1)?¿y LR(1)?

Justificar convenientemente las respuestas.

3.- Existen algunos lenguajes (entre ellos el FORTRAN primitivo), en los que se permite la definición y utilización de funciones, pero no la invocación recursiva de éstas (ni directa ni indirectamente); por otro lado, en estos lenguajes no existe el concepto de manejo dinámico de la memoria (*free-malloc* ó *NEW-DISPOSE*). ¿Qué estructura de memoria en tiempo de ejecución sería la más conveniente para un lenguaje con tales características?

Justificar convenientemente la respuesta.

4.- Volviendo a la gestión de tipos complejos vista en clase (**POINTER TO**, **ARRAY OF**, etc.), y considerando la construcción de tipos complejos, la declaración de variables de tales tipos, y la asignación de expresiones en las que intervienen modificadores de tipos (^, **[]**, etc.), responder a la siguiente cuestión:

¿Qué modificaciones habría que hacer, desde el punto de vista sintáctico y semántico, para permitir el uso de tipos definidos por el usuario? Ej.:

```
TYPE
      miPuntero = POINTER TO INTEGER;
      miArray = ARRAY OF miPuntero;
VAR
      uno : miPuntero;
      dos, tres : miArray;
      cuatro : POINTER TO POINTER TO PROCEDURE (): REAL;
BEGIN
      /* Igual que lo visto en clase */
END
```

Justificar convenientemente la respuesta.

PRÁCTICA

Se desea reconocer el lenguaje SUBASIC, aceptado por la siguiente gramática:

```
programa    :     /* Epsilon */
            |     programa sent
            |     programa error
            ;
            ;
sent        :     IF cond THEN programa END IF
            |     FOR ID '=' expr TO expr DO programa NEXT ID
            |     WHILE cond DO programa END DO
            |     SKIP
            |     CONTINUE
            |     PRINT CADENA
            |     PRINT expr
            |     PRINTLN CADENA
            |     PRINTLN expr
            |     lvalor '=' expr
            ;
            ;
lvalor      :     ID
            |     ID '[' expr ']'
            ;
            ;
expr        :     lvalor
            |     NUM
            |     expr '+' expr
            |     expr '*' expr
            |     expr '-' expr
            |     expr '/' expr
            |     '-' expr
            ;
            ;
cond        :     expr COMP expr
            |     cond OR cond
            |     cond AND cond
            |     NOT cond
            ;
            ;
```

El siguiente programa representa un ejemplo de entrada válido para esta gramática:

```
FOR I=1 TO 100 DO
      FOR J=1 TO 10 DO
            IF A[I]==0 OR A[J]==0 THEN
                  SKIP
            END IF
            CONTADOR=0
            WHILE A[J]<>0 DO
                  A[J]=A[J] - 1
                  CONTADOR=A[I]+CONTADOR
                  A[I]=A[I]+CONTADOR
            END DO
            A[J]=A[I]*CONTADOR
      NEXT J
      PRINT "Esto es un ejemplo de PRINT"
      PRINTLN A[I]
NEXT I
PRINT "Se acabo"
```

Este lenguaje va a ser interpretado en máquinas de reducida capacidad, por lo que se desea construir un conversor que codifique un programa fuente (**que supondremos válido sintácticamente**) y lo reduzca en tamaño. Para convertir un programa fuente en un programa fuente codificado se utilizarán las siguiente reglas:

a) Cada palabra reservada se sustituirá por un carácter ASCII según la siguiente tabla:

Pal. reservada	Cód. ASCII

IF	0x01
THEN	0x02
END	0x03
FOR	0x04
TO	0x05
NEXT	0x06
WHILE	0x07
DO	0x08
SKIP	0x09
CONTINUE	0x0A
PRINT	0x0B
PRINTLN	0x0C

AND	0x0D
OR	0x0E
NOT	0x0F
= =	0x11
<>	0x12
<=	0x13
>=	0x14
<	0x15
>	0x16

donde p.ej. 0x07 representa el carácter cuyo código ASCII es el 07 en hexadecimal.

b) Las cadenas de caracteres se transforman de la misma forma en que se encuentran (incluidas las comillas dobles).

c) Un texto que representa a un número entero se codifica mediante dos códigos ASCII siguiendo el modelo *little-endian*. Ej.:

"127" (que equivale a 0x007F) se almacena como los caracteres: 0x00 0x7F.

"12965" (que equivale a 0x32A5) se almacena como los caracteres: 0x32 0xA5.

Para evitar confusiones, toda número entero codificad irá precedido del código ASCII 0x10.

d) Los nombres de las variables se codifican usando el siguiente criterio

d.1) La primera vez que aparece una cierta variable, ésta se sustituye por un código numérico incremental seguido del nombre de la variable: a la primera variable se le asocia el código 1, a la segunda el código 2, a la tercera el 3, y así sucesivamente. Se permite un máximo de 2^{16}-1 variables distintas, por lo que el código se almacenará como un valor natural siguiendo el modelo *little-endian* descrito anteriormente. Para saber cuándo acaba un nombre de variable, se le pone el código ASCII 0x17 detrás de su último carácter.

d.2) La segunda y sucesivas veces que aparece una cierta variable, ésta se sustituye únicamente por el código que se le asoció en el punto anterior.

d.3) Para evitar confusiones, toda variable codificada irá precedida por el código ASCII 0x17.

e) Los espacios, tabuladores y retornos de carro se ignoran.

f) Cualquier otro carácter se almacena como sí mismo (su propio código ASCII).

Para aclarar todo este proceso, la siguiente tabla ilustra el volcado hexadecimal del resultado de convertir el programa fuente anterior; la columna central son los códigos ASCII que integran el programa fuente codificado; la columna de la derecha asocia a cada código ASCII el carácter ASCII que le corresponde.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	04	17	00	01	49	17	3D	10	00	01	05	10	00	64	08	04	....I.=.....d..
00000010	17	00	02	4A	17	3D	10	00	01	05	10	00	0D	0A	08	01	...J.=..........
00000020	17	00	03	41	17	5B	17	00	01	5D	11	10	00	00	0E	17	...A.[...]......
00000030	00	03	5B	17	00	02	5D	11	10	00	00	02	09	03	01	17	..[...].........
00000040	00	04	43	4F	4E	54	41	44	4F	52	17	3D	10	00	00	07	..CONTADOR.=....
00000050	17	00	03	5B	17	00	02	5D	12	10	00	00	08	17	00	03	...[...]........
00000060	5B	17	00	02	5D	3D	17	00	03	5B	17	00	02	5D	2D	10	[...]=...[...]-.
00000070	00	01	17	00	04	3D	17	00	03	5B	17	00	01	5D	2B	17	.....=...[...]+.
00000080	00	04	17	00	03	5B	17	00	01	5D	3D	17	00	03	5B	17	.....[...]=...[.
00000090	00	01	5D	2B	17	00	04	03	08	17	00	03	5B	17	00	02	..]+........[...
000000A0	5D	3D	17	00	03	5B	17	00	01	5D	2A	17	00	04	06	17	]=...[...]*.....
000000B0	00	02	0B	22	45	73	74	6F	20	65	73	20	75	6E	20	65	..."Esto es un e
000000C0	6A	65	6D	70	6C	6F	20	64	65	20	50	52	49	4E	54	22	jemplo de PRINT"
000000D0	0C	17	00	03	5B	17	00	01	5D	06	17	00	01	0B	22	53	....[...]......"S
000000E0	65	20	61	63	61	62	6F	22									e acabo"

Se pide construir los programas Lex/Yacc que sean necesarios para producir el comportamiento que se acaba de explicar.

Nota: Reflexionar sobre si es necesario hacer o no un programa Yacc y si es suficiente tan sólo un programa Lex para resolver el problema, (justificar la respuesta convenientemente).