PROCESADORES DE LENGUAJE (LABORATORIO) *Junio-2010*
4º Curso Ingeniería Informática Tiempo: 2 horas
Nombre: Apellidos: Grupo:

(3 ptos) Pregunta 1:

Dada la instrucción INSERT de SQL:

```
INSERT INTO Tabla (campo1, campo2, ..., campoN)
VALUES (valor1, valor2, ..., valorN)
```

Se pide:

a. (1 ptos) Diseña una gramática LL(1) que permita generar instrucciones de tipo INSERT en SQL.

b. (1.5 ptos) Implementa todos los procedimientos según el método de análisis descendente recursivo predictivo. Y en particular, para el procedimiento que genere la lista de valores incluye la recuperación en modo de pánico. En caso de error, se debe indicar el *token* que se ha encontrado y lo que se esperaba en ese momento.

c. (0.5 ptos) Añade las acciones semánticas necesarias para que el número de campos y el número de valores sea el mismo. En caso contrario, emite un mensaje de error.

(3 ptos) Pregunta 2:

Dada la siguiente gramática extraída de la práctica 2 donde se ha considerado sólo la parte de declaraciones de funciones:

```
Programa → DeclaracionFuncs DefEscenario
DeclaracionFuncs  →  FUNCIONES  ID ( ArgObjetos ) ListaDeclFuncs | ε
ArgObjetos →  ID ListaObjetos | ε
ListaObjetos → , ID ListaObjetos  | ε
ListaDeclFuncs → ID ( ArgObjetos ) ListaDeclFuncs | ε
DefEscenario → ESCENARIO INICIO Sentencia ListaSents FIN
Sentencia → ID ( ArgObjetos )
```

a. (0,5 ptos.) Escribe la **estructura** de tabla de símbolos que vas a necesitar aquí (y que será una versión reducida de la utilizada en la práctica) y los **prototipos** de las funciones para el manejo de la tabla de símbolos.

b. (1,5 ptos.) Inserta en la gramática las acciones necesarias para rellenar la tabla de símbolos con la información sobre las funciones utilizando las variables $. Se debe incluir: el número y el nombre de los parámetros.

c. (1 ptos.) En las llamadas a funciones deberemos comprobar que esa función ha sido definida y que el número de argumentos que se le pasa es el correcto. Implementa esta comprobación.

NOTA: No hay que implementar las funciones de manejo de la tabla de símbolos, únicamente sus prototipos.

(4 ptos) Pregunta 3:

Dada la versión muy reducida de la gramática de la práctica 3:

```
Programa → DeclaracionObjetos DefEscenario
DeclaracionObjetos → OBJETOS DeclObjetos ListaDeclObjetos
ListaDeclObjetos → DeclObjetos ListaDeclObjetos | ε
DeclObjetos → ID ListaObjetos NVERTICES NUM
ListaObjetos → , ID ListaObjetos | ε

DefEscenario → INICIO Sentencia ListaSents FIN
ListaSents → Sentencia ListaSents | ε
Sentencia → Dibujar ID                    |   /* Dibuja el objeto indicado*/
          Si Condicion { Sentencia ListaSents } |
          Si Condicion { Sentencia ListaSents }
          Sino { Sentencia ListaSents }
Condicion → Dato == Dato
Dato → NUM | ID
```

a) (1,0 ptos.) Compacta la gramática según la notación EBNF.
b) (1,0 ptos.) Introduce los operadores para la creación del árbol sintáctico. Dibuja el árbol que obtendrías para el siguiente ejemplo.

NOTA: Los apartados a y b se realizarán conjuntamente, se ha dividido para indicar el valor de cada parte usando la estructura de PCCTS (la clase MiParser).

OBJETOS
cubo, cubo2 NVERTICES 8
rombo NVERTICES 4
INICIO
Dibujar cubo
Si a == 3 {
Dibujar rombo
}
Sino {
Dibujar cubo2
}
FIN
	DEF_CAB INICIO 0
	DRAW cubo
	STORE t0 3
	IS_EQUAL t1 a t0
	NOT t2 t1
	GOTO_REL t2 3
	DRAW rombo
	GOTO_REL t1 2
	DRAW cubo2

c) (2 ptos.) Implementa toda la función `genera_codigo` en PCCTS de acuerdo al árbol dibujado de forma que saque el siguiente código intermedio (el mismo que en la práctica excluyendo la creación de página web):

LISTADO DE OPERADORES PARA GENERAR LOS CUADRUPLOS

DEF_CAB TKN_ID TKN_NUM //define la cabecera con TKN_ID y el numero de argumentos TKN_NUM

DRAW TKN_ID //Dibuja el objeto especificado en el TKN_ID

STORE TKN_ID TKN_NUM //Guarda en TKN_ID (vble. Temporal) el nºTKN_NUM

IS_EQUAL ID_TKN ID_TKN ID_TKN // *id1 (id2 == id3)*

GOTO_REL TKN_ID TKN_NUM //salto relativo: si la var. ID es verdadera
//(diferente de 0) se salta el número de líneas indicado en NUM

NOT TKN_ID TKN_ID *// id1 NOT id2*