PROCESADORES DE LENGUAJE (LABORATORIO) *Septiembre-2005*
4º Curso Ingeniería Informática Tiempo: 90 minutos
No se permiten ni libros ni apuntes.

(1,5 p.) Diseñar una gramática LL(1) que genere sentencias de llamadas a funciones con un número indeterminado de argumentos. Asumid que:

- Puede existir anidamiento en llamadas a funciones, es decir, un argumento puede ser lo que devuelva la llamada a una función.

- Pueden existir funciones sin argumentos.

- Sólo tenemos la operación asignación.

Ejemplo:

```
a = Suma(Suma(a,b), c)
```

(4 p.) A partir del siguiente extracto de la gramática de la práctica 2

```
Programa          → Subrutinas Bloque
Subrutinas        → Funcion Subrutinas | Procedimiento Subrutinas | ε
Procedimiento     → TKN_SUB TKN_ID TKN_PAA Argumentos Bloque TKN_SUB
Funcion           → TKN_FUNCTION TKN_ID TKN_PAA Argumentos TKN_AS Tipo Bloque TKN_FUNCTION
Argumentos        → TKN_ID _TKN_AS Tipo MasArgumentos | TKN_PAC
MasArgumentos     → TKN_COMA  TKN_ID_TKN_AS Tipo MasArgumentos | TKN_PAC
Tipo              → TKN_INTEGER | TKN_SINGLE | TKN_STRING
Bloque            →  ... (para este ejercicio no hace falta)
```

Resuelve por separado las siguientes cuestiones:

a) Diseña un pequeño programa en Bison que obtenga el número de argumentos (en total) y cuantos hay de cada tipo (integer, single y string).

b) Diseña la estructura de la Tabla de Símbolos e introduce las acciones necesarias para controlar que:

- no existen dos variables con el mismo nombre y ámbito.
- los nombres de las funciones y procedimientos son únicos (no existe ninguna otra función ni procedimiento, ni variable de cualquier ámbito con ese nombre).

(3,5 p.) Dada la siguiente gramática:

```
Bloque        →  Instruccion TKN_ PTOCOMA MasInstruccion
Instruccion   → TKN_ID TKN_IGUAL Expresion | TKN_PRINT Impresion | TKN_LEN TKN_PAA TKN_ID TKN_PAC
MasInstruccion → Instruccion TKN_PTOCOMA MasInstruccion  | ε
Impresion     →  TKN_ID MasImp | Expresion MasImp
MasImp        → TKN_COMA Impresion MasImp | ε
Expresion     →     Expresion  TKN_MAS  Expresion  |
                    Expresion  TKN_MENOS  Expresion  |
                    Expresion  TKN_POR  Expresion  |
                    Expresion  TKN_DIV  Expresion  |
                    TKN_ID  |  TKN_NUMENT | TKN_NUMREAL
```

PROCESADORES DE LENGUAJE (LABORATORIO) *Septiembre-2005*
4º Curso Ingeniería Informática Tiempo: 90 minutos
No se permiten ni libros ni apuntes.

Se pide:

- **Compactarla y crear el árbol sintáctico haciendo uso de la herramienta del PCCTS.**
- **Dibujar cuál sería la salida que construiría el PCCTS, recorrido en preorden, para la siguiente entrada:**

```
PRINT cad , num ;
num = 3*4+5;
PRINT 5*2+1 , cad;
```

- **Implementa las partes de la función "genera_codigo" que nos permite generar el siguiente código intermedio:**

```
y = x + 2;

# el codigo generado sería
ASSIG_C     t0    2     //guarda el numero 2 en la variable t0
ADD         t1    x     t0    // sumar los elementos y los guarda en t1
ASSIG_V     y     t1          // copia t1 en y
```

(1,0 p.) Dada la clase AST del PCCTS, implementa la función preorder que recorre el árbol en forma prefija y lo imprime.

```
class AST : public ASTBase {
protected:
      ANTLRTokenPtr token;
public:
      AST(ANTLRTokenType tok, char *s) { token = new ANTLRToken(tok, s); }
      AST(ANTLRTokenPtr t) { token = t; }
      void preorder_action() {
            char *s = token->getText();
            cout <<  s  ; }
       void preorder_before_action() {
            cout <<  "(" ; }
       void preorder_after_action() {
            cout <<  ")"  ; }
      char *getText()        { return token->getText(); }
      void preorder( )
      {
            …… (FUNCIÓN A IMPLEMENTAR)
       }
    }
```