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 Diciembre de 2008
y ordinaria de Febrero de 2009
3er Curso de I.T. Informática de Sistemas.

Apellidos, Nombre: _____
Calificación: _____

Se desea construir una pequeña calculadora que permite utilizar las operaciones básicas de suma, producto, módulo y complemento a 1. Esta calculadora permite tratar valores enteros positivos y permite el uso de variables formadas por una sola letra minúscula.

A pesar de su simplicidad, la calculadora posee las siguientes particularidades:

- Una constante entera puede expresarse en base 10 (por defecto), en base 2, en base 8 o en base 16. En los tres últimos casos se le coloca la base como sufijo del número y separada de éste por un abre paréntesis. Ejs.:
  - 30 representa el entero de valor 30 en decimal.
  - 1001(2 representa el entero de valor 9 en decimal.
  - 5446(8 representa el entero de valor 2854 en decimal.
  - 1FA(16 representa el entero de valor 506 en decimal.
- La asignación es simple y está formada por la palabra LET seguida de la variable que va a recibir el valor, el signo igual y la expresión cuyo valor se desea asignar. Ejs.:

```
LET a = 30;
LET a = a + FF(16 % 10(16;
```

asigna a la variable a el valor 30 y luego el valor 45.

- El valor entero de una expresión cualquiera puede visualizarse por pantalla mediante la sentencia PRINT. Esta sentencia puede ir seguida de dos partes opcionales: una que especifica la base en que se quiere visualizar el valor, y otra que indica la condición que debe cumplirse para que se visualice o no dicho valor. Ejs.:

```
PRINT 1*2+1;
PRINT a;
PRINT a AS HEXADECIMAL;
PRINT a IF a > 0;
PRINT a AS BINARY IF a > 0;
```

- Las únicas condiciones que se permiten son las que comparan dos valores mediante el operador relacional mayor que (>), y las que emplean los operadores lógico AND y NOT.

Así, la siguiente tabla muestra a la derecha la salida que debe emitirse por pantalla ante la entrada de la izquierda:

Entrada	Salida
LET a = 30; PRINT a; PRINT a AS HEXADECIMAL; PRINT a AS OCTAL; PRINT a AS BINARY;	 30 1e 36 11110

PRINT COMPLEMENT(a) AS BINARY; LET b = 1001(2; PRINT b; LET c = 5446(8; PRINT c; LET d = 1FA(16; PRINT d; PRINT 1 +1 * 2; PRINT 1*2+1; PRINT FF(16 % 10(16 + a; PRINT a + FF(16 % 10(16; LET a = a + FF(16 % 10000(2; PRINT a IF a > 0; PRINT a AS BINARY IF a > 0; PRINT 01101001(2; PRINT a IF 0 > a;	11111111111111111111111111100001 9 2854 506 3 3 45 45 45 101101 105

Se pide:

- Construir los programas Lex y YACC que proporcionen la funcionalidad solicitada. Para ello se proporcionan los siguientes elementos:

1) Un esqueleto de Lex que incluye tres funciones que permiten convertir a enteros decimales cadenas que representan números en distintas bases.

2) Un esqueleto de YACC con la gramática ya creada así como una tabla de símbolos formada por un array de enteros.

3) La función itoa(int valor_decimal, char * cadena, int base). Esta función transforma un valor entero en su representación en forma de cadena de caracteres, representación que se almacena en el segundo parámetro (que suele ser un array de char previamente definido). Además, permite especificar en qué base queremos representar el resultado.

Exdic08l.lex

```
%{
	// Función que transforma un texto formado por 0s y 1s en su valor entero asociado.
	int deBinario(char * texto);
	// Función que transforma un texto expresado en octal a su valor entero asociado.
	int deOctal(char * texto);
	// Función que transforma un texto expresado en hexadecimal a su valor entero asociado.
	int deHexadecimal(char * texto);
%}
%%
```

Exdic08y.yac

```
%{
#include <stdio.h>
#include <string.h>
 // Definir aquí el tipo de los atributos (todos son de tipo int).

int vbles['z' - 'a' + 1];
%}
 // Definir aquí los tokens y las precedencias


%%
```

```
prog :      /* Epsilon */
        |   prog sent ';'
        |   prog error ';' {          }
        ;
sent :      LET ID '=' expr          {              }
        |   PRINT expr opc1 opc2     {
                                     }
        ;
opc1 :      /* Epsilon */        {              }
        |   AS BINARY            {              }
        |   AS OCTAL             {              }
        |   AS DECIMAL           {              }
        |   AS HEXADECIMAL       {              }
        ;
opc2 :      /* Epsilon */ {                 }
        |   IF cond       {                 }
        ;
expr :      NUM                          {                   }
        |   ID                           {                   }
        |   expr '+' expr                {                   }
        |   expr '*' expr                {                   }
        |   '(' expr ')'                 {                   }
        |   expr '%' expr                {                   }
        |   COMPLEMENT '(' expr ')'      {                   }
        ;
cond :      expr '>' expr                {                   }
        |   cond AND cond                {                   }
        |   NOT cond                     {                   }
        |   '(' cond ')'                 {                   }
        ;
%%
#include "errorlib.c"
#include "exdic08l.c"
int main(){
        int i;
        for(i='a'; i<='z'; i++)
                vbles[i - 'a'] = 0;
        yyparse();
        return 0;
}
```