Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

PRUEBA T1

Traductores, compiladores e intérpretes

Septiembre. Curso 08/09

Nombre y apellidos: _____

Grupo (mañana - tarde). Táchese lo que no proceda.

Se desea construir un programa Lex que reconozca los siguientes patrones y realice las acciones indicadas:

1. Espacios, tabuladores que deben ignorarse.
2. Retornos de carro que sirven para contar el número de líneas. Por lo demás deben ignorarse.
3. Comentarios que pueden estar anidados. Deben ignorarse. Un comentario empieza por (* y finaliza en *).
4. Las palabras reservadas LOOP, EXIT, WHEN y END. Deben ignorarse.
5. Identificadores de usuario que sólo pueden estar formados por una letra minúscula. Debe contarse cuántas veces aparece cada identificador. Al final del análisis debe aparecer el total de identificadores encontrados.
6. Cualquier otra cosa que se reconozca debe emitirse por pantalla avisando de que se trata de un error. Además hay que indicar el número de línea en que se encuentra el fallo.

Al finalizar la función yylex() debe indicarse por pantalla:

- El número de líneas de la entrada reconocida.
- El número de palabras reservadas reconocido.
- El número de veces que aparece cada identificador.

Un ejemplo de entrada y salida sería:

Entrada	Salida
(* Un ejemplo *) LOOP , WHEN abcaab EEND a a a (* (* a a a (* aaa (* END LOOP FOR cualquier cosa *) *) b b *) A *) A % IF b cc (* FOR *)	 Fallo en linea 2 con ,. Fallo en linea 6 con E. Fallo en linea 13 con A. Fallo en linea 14 con %. Fallo en linea 15 con I. Fallo en linea 15 con F. No. de lineas: 16. No. de palabras reservadas: 3. a aparece 6 veces. b aparece 3 veces. c aparece 3 veces.

Universidad de Málaga
Departamento de Lenguajes y
Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

PRUEBA T2

Traductores, compiladores e intérpretes

Septiembre. Curso 08/09

Nombre y apellidos: _____

Grupo (mañana - tarde). Táchese lo que no proceda.

Se desea construir los programas Lex y YACC destinados a reconocer como válidas listas de números naturales pares separados por comas que constituyan secuencias no descendentes. Una secuencia debe estar formada por, al menos, tres números de manera que cada número (del segundo en adelante) debe ser mayor o igual a la suma de todos los anteriores. Por ejemplo:

Secuencia	¿Es válida?
2, 2, 6, 10, 22, 50	Sí
1, 1, 2, 3, 7, 10, 17	No
0, 0, 11, 12, 23, 35, 58	No
2, 4, 10, 16, 30, 100	No
0, 0, 0, 0, 0, 0, 0	Sí
4, 6	Error de sintaxis

El programa, tras leer la secuencia, debe emitir por pantalla sencillamente la palabra SÍ o NO indicando, respectivamente, si la secuencia leída es o no válida.

Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

PRUEBA T3
Traductores, compiladores e intérpretes
Septiembre. Curso 08/09

Nombre y apellidos: _____
Grupo (mañana - tarde). Táchese lo que no proceda.

En este ejercicio se parte del siguiente supuesto:
Un amigo que estudia Ciencias de la comunicación nos ha proporcionado una película muda junto con un fichero de texto con los subtítulos en español. El fichero de subtítulos tiene algunos problemas que hacen que no sea reconocido bien por nuestro reproductor y que la aparición de subtítulos salga a destiempo con respecto a la imagen de la pantalla. Nuestro objetivo será solucionar esos problemas.

El fichero de subtítulos estándar (extensión **.srt**) tiene el siguiente formato del siguiente ejemplo:

```
1
00:00:20,000 --> 00:00:24,400
El Estado se burla del ciudadano
y se adjudica el derecho a decidir por el,

2
00:00:24,600 --> 00:00:27,800
tan solo porque es votado cada lustro...
```

En este ejemplo puede observarse cómo cada subtítulo está formado por, al menos, dos líneas, siendo la primera un número consecutivo de 1 en 1 empezando por el 1 y la segunda un lapso de tiempo donde el inicio y el final están expresados en hh:mm:ss:eee (horas, minutos, segundos y milésimas) y separados por el símbolo "-->". A partir de esta segunda línea puede haber una o más líneas que son el contenido en sí del subtítulo, de manera que la idea es que dicho texto aparezca en pantalla durante el lapso de tiempo indicado por la línea que le precede.

En nuestro caso debemos leer un fichero con estas características controlando lo siguiente:

- En la entrada, cada bloque de subtítulo puede estar separado por varias líneas en blanco. Nosotros debemos separarlas por sólo una.
- Si un bloque de subtítulo no tiene línea/s de texto asociada, nosotros no la emitiremos y sacaremos un mensaje de error.
- Debemos asegurarnos que los números de los subtítulos sean incrementales de uno en uno partir del 1. Si algún subtítulo no cumple esto sacaremos un mensaje de error y no lo emitiremos.
- Debemos asegurarnos que el primer valor de un lapso de tiempo es inferior al segundo valor. Si algún subtítulo no cumple esto sacaremos un mensaje de error y no lo emitiremos.
- Debemos asegurarnos que el primer valor de un lapso de tiempo es superio al segundo valor del lapso anterior. Si algún subtítulo no cumple esto sacaremos un mensaje de error y no lo emitiremos.
- Ante cualquier error de sintaxis sacaremos un mensaje de error y no emitiremos el subtítulo que tenga el error.
- A todos los valores de tiempo hay que añadirles 12 segundos y 630 milésimas para sincronizarlos con la película.

Con este objetivo en mente y partiendo de los programas LEX y YACC que se proporcionan, se pide completarlos para conseguir el propósito del enunciado..
Nota: Se recomienda el uso de la función **sprintf**.

Como ejemplo, se muestra la salida que debe producirse ante una cierta entrada:

Entrada	***Salida***
1 00:00:20,000 --> 00:00:24,400 El Estado se burla del ciudadano y se adjudica el derecho a decidir por el, 2 00:00:24,600 --> 00:00:27,800 tan solo porque es votado cada lustro. 3 00:00:30,600 --> 00:00:37,800 Se jactan de "democratas" cuando realmente son "demofobos". 4 00:00:37,801 --> 00:00:37,802 5 00:00:40,600 --> 00:00:37,800 El alcalde es nuestro servidor. 6 00:00:50,600 --> 00:00:57,800 La palabra "ministro" procede del latin "minister" y quiere decir servidor. 7 00:00:20,000 --> 00:00:24,400 ¿Aun crees que decides lo más minimo excepto quien te roba? 8 00:00:60,600 --> 00:00:69,000 FIN	1 00:00:32,630 --> 00:00:37,030 El Estado se burla del ciudadano y se adjudica el derecho a decidir por el, 2 00:00:37,230 --> 00:00:40,430 tan solo porque es votado cada lustro. 3 00:00:43,230 --> 00:00:50,430 Se jactan de "democratas" cuando realmente son "demofobos". Falta el texto del subtitulo. Lapso de tiempo incorrecto. 6 00:01:03,230 --> 00:01:10,430 La palabra "ministro" procede del latin "minister" y quiere decir servidor. Lapso de tiempo no consecutivo con el anterior. 8 00:01:13,230 --> 00:01:21,630 FIN

Fichero EJEM9L.LEX

```
%{
int actual=0;
%}
%%
^[1-9][0-9]*   {
                    yylval.numero = atoi(yytext);

                    return NUMERO;
                }
[0-9][0-9]\:[0-9][0-9]\:[0-9][0-9]\,[0-9][0-9][0-9]   {
                    yytext[2]=yytext[5]=yytext[8]=0;
                    yylval.tiempo.hora = atoi(yytext);
                    yylval.tiempo.minuto=atoi(yytext+3);
                    yylval.tiempo.segundo=atoi(yytext+6);
                    yylval.tiempo.milesima=atoi(yytext+9);
                    return TIEMPO;
                                                        }
"-->"       { return FLECHA; }
[^ \t\n]+   { strcpy(yylval._texto, yytext); return PALABRA; }
[ \t]+      { return ESPACIO; }
\n          { return '\n'; }
```

Fichero t3y.yac

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define false 0
#define true 1
int fallo=false;
long finLapsoAnterior=0; // Valor expresado en milésimas;
typedef struct {
   int hora, minuto, segundo, milesima;
} _tiempo;
void incrementa(_tiempo * ptr){
   ptr->milesima+=630;
   // Controlar el acarreo y sumar los 12 segundos.

}
void borraRetornosAlFinal(char * texto){
   int i;
   for(i=strlen(texto)-1; i!=-1; i--)
      if (texto[i]==10 || texto[i]==13)
         texto[i]=0;
      else
         return;
}
%}
%union{
   int numero;
   _tiempo tiempo;
   char _texto[200];
}
%token ESPACIO FLECHA
%token <texto> PALABRA
```

```
%token <numero> NUMERO
%token <tiempo> TIEMPO
%type <texto> texto lapso
%%
prog :   retornosIniciales
    |  prog NUMERO '\n' lapso '\n' texto {
                                  if (! fallo){




                                  }
                                  fallo = false;
                                 }
    | prog error
    ;
retornosIniciales    : /* Epsilon */
                | retornosIniciales '\n' ;
lapso : TIEMPO  ESPACIO FLECHA ESPACIO TIEMPO {
                long t1, t2;
                t1 = ((long)$1.hora)*3600000L+((long)$1.minuto)*60000L+((long)$1.segundo)*1000L+((long)$1.milesima);
                t2 = ((long)$5.hora)*3600000L+((long)$5.minuto)*60000L+((long)$5.segundo)*1000L+((long)$5.milesima);
                // Controlar el fin del lapso anterior
                // Controlar que el lapso sea creciente









                                                    }
    ;
texto    : /* Epsilon */   {                    }
      | texto PALABRA {                              }
      | texto ESPACIO {                            }
      | texto TIEMPO    {

                          }
      | texto '\n'   {                                                          }
      ;
%%
#include "t3l.c"
int main(){
   yyparse();
   return 0;
}

void yyerror(char * s){
   printf("%s\n", s);
}
```