PROGRAMA 1. ANALIZADOR LEXICO. COMPILADORES E INTÉRPRETES.

En este primer programa deben implementar el analizador léxico para el lenguaje que se muestra en la parte inferior de este documento. El analizador léxico debe ser implementado a nivel modular, es decir, dependiendo del lenguaje de programación que seleccionen, un método o función debe ser el encargado de regresar el siguiente token que se encuentre en el código fuente que haya sido pasado como argumento al programa. Tal método ó función debe regresar un entero que identifique al siguiente token encontrado y en su caso debe también regresar un apuntador al lexema encontrado. También debe mantener el número de línea actual para que en caso de error se pueda llamar a una función ó método que imprima el mensaje de error que indique tanto el tipo de error como el número de línea en que se encuentra (para obtener 10 puntos extra también puede indicar en que parte de la línea esta el error). Recuerden que los únicos errores que un analizador léxico puede encontrar son en cuanto a tokens mal formados:

- Numero Entero No Valido (por ejemplo: 67fr)
- Identificador No Valido (por ejemplo: a$2d)
- Elemento No Valido (por ejemplo el carácter :)
- Archivo No Existente (cuando el archivo indicado no se puede encontrar ó no se puede abrir)

Los identificadores deben comenzar con letra y deben ser seguidos por letras, dígitos o subrayados. No hay diferencia entre mayúsculas y minúsculas y pueden tener una extensión de hasta 32 caracteres.

En el lenguaje solo hay números enteros (no hay reales).

Las constantes enteras y los identificadores deben ser almacenados en una tabla de símbolos (asegúrese de declarar la tabla de símbolos de tal manera que pueda de manera fácil agregarle posteriormente atributos, por el momento es sólo necesario el nombre y tipo.

A cada uno de los tipos de componentes se le asigna un entero que identifica el tipo de acuerdo a los siguientes valores:

Token	**Número**
CONST	1
VAR	2
PROC	3
INICIO	4
FIN	5
ESCRIBIR	6
LEER	7
LLAMAR	8
SI	9
MIENTRAS	10

PARA	11
(id)Identificador	12
(num) NumeroEntero	13
.	14
=	15
,	16
;	17
(	18
)	19
&	20
%	21
==	22
#	23
<	24
>	25
<=	26
>=	27
+	28
-	29
*	30
/	31

Los comentarios pueden ser multilíneas (entre { y }) ó de una sola línea (todo lo que va desde // en una línea hasta el final de la misma) y deben ser omitidos, así como los espacios en blanco.

El programa principal se ejecutará de la siguiente manera:

```
Compilador nombre_archivo
```

donde *nombre_archivo* es el nombre del archivo de texto que contiene el código fuente a compilar.

Así, si por ejemplo el archivo prog1.pl0 contiene las siguientes líneas:

```
{ ESTE PROGRAMA ES UNA MUESTRA
   DEL LENGUAJE PL/0
}
CONST MAX = 5; // Esta es una constante
VAR num,
    suma;

INICIO
    LEER num;
      SI (num < MAX)
```

```
        INICIO
            suma = num * (10 + 7) / 2;
            ESCRIBIR suma
        FIN
FIN.
```

y se corre el programa de la siguiente manera:

```
Compila prog1.pl0
```

el programa principal deberá llamar a la función de análisis léxico que obtiene el siguiente token hasta que el archivo fuente termine imprimiendo cada uno como sigue:

```
#TOKEN          LEXEMA
1
12              MAX
15
13              5
17
2
12              num
16
12              suma
17
4
7
12              num
17
9
18
12              num
24
12              MAX
19
4
12              suma
15
12              num
30
18
13              10
28
13              7
19
31
13              2
17
6
12              suma
5
5
14
```

En cambio si el archivo prog1.pl0 hubiera contenido el siguiente código:

```
{ ESTE PROGRAMA ES UNA MUESTRA
   DEL LENGUAJE PL/0
}
CONST MAX = 5; // Esta es una constante
VAR num,
```

```
    suma$;

INICIO
    LEER num;
        SI (num < MAX)
          INICIO
             suma = num * (10 + 7) / 2;
                 ESCRIBIR suma
          FIN
FIN.
```

la salida sería la siguiente:

```
1
12              MAX
15
13              5
17
2
12              num
16
prog1.pl0: Error en linea 6 : Identificador No Valido
    suma$;
        ^
```

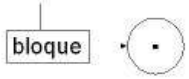
NOTE QUE LA SALIDA DEBE SER TAL COMO SE MUESTRA (ENTRE CADA COLUMNA HAY DOS TABULADORES, ¡NO SON ESPACIOS!) YA QUE SE VA A CALIFICAR DE MANERA AUTOMATICA EN BASE A LA SALIDA Y SI NO COINCIDE SE CONSIDERARA QUE EL PROGRAMA NO FUNCIONA COMO SE ESPERA.

RECUERDEN QUE LOS TRABAJOS DE PROGRAMACION NO SON REEMPLAZABLES LO CUAL SIGNIFICA QUE SI NO SON ENTREGADOS NO HABRA FORMA DE RECUPERAR EL PORCENTAJE DE LA CALIFICACION FINAL QUE LES CORRESPONDE. EL PROGRAMA PUEDE SER HECHO EN EL LENGUAJE DE PROGRAMACION DE SU PREFERENCIA. COMO COMENTARIO EN EL PROGRAMA DEBE ESTAR INDICADO SU NOMBRE Y MATRICULA (EL PROGRAMA PUEDE SER REALIZADO DE MANERA INDIVIDUAL O EN EQUIPOS DE DOS PERSONAS)

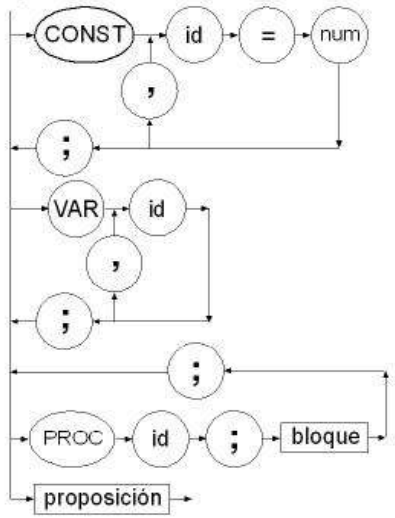
Por favor envíe su archivo fuente usando la liga que se encuentra en la página del programa a más tardar a las 11:55 PM del martes 14 de marzo del 2017. El programa fuente (y todos los archivos que este necesite) deben ser empacados en un archivo .zip o .rar que es el que debe de enviar usando la liga de abajo. El nombre que utilice para las variables y funciones en su código deben ser significativos, es decir, deben dar alguna idea sobre lo que están representando, si usa identificadores tales como xxx, xya, o similares el programa se le asignará una calificación reprobatoria. Finalmente, si se reciben programas copiados, todas las personas que así lo entreguen recibirán calificación reprobatoria EN LA MATERIA, NO SOLAMENTE EN EL PROGRAMA. RECUERDEN QUE LOS PROGRAMAS SERAN REALIZADOS DE MANERA INDIVIDUAL O EN EQUIPOS DE DOS PERSONAS Y TODA PERSONA QUE LOS ENTREGUE DEBE CONOCER EL FUNCIONAMIENTO DE LOS MISMOS, SI SE LES PREGUNTA RESPECTO DEL MISMO EN UN EXAMEN O DE MANERA ORAL Y NO PUEDE RESPONDER EL PROGRAMA SERA CONSIDERADO COMO COPIADO.

DIAGRAMAS SINTACTICOS DE PL/O

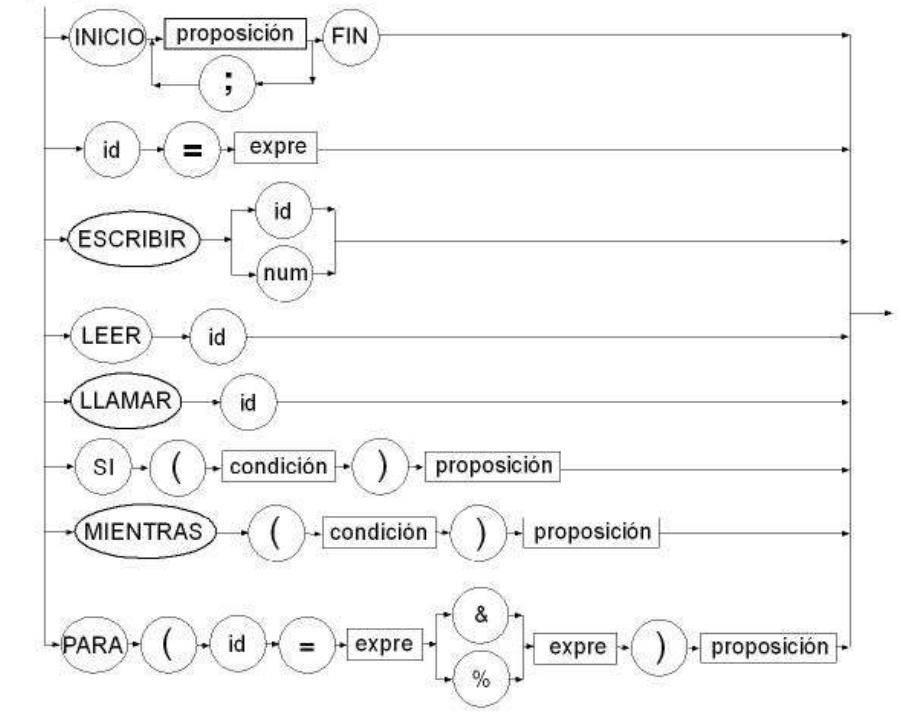
<programa>



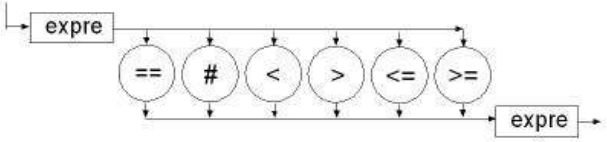
<bloque>



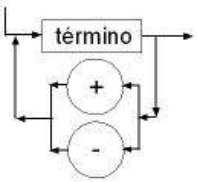
<proposición>



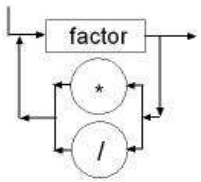
<condición>



<expre>



<término>



<factor>