Trabajo Práctico N.º 3

En base a la gramática léxica hecha en el TP1 armar un programa que la reconozca usando la herramienta flex. Tenga en cuenta que los tokens devueltos nos son directamente los documentados, sino que puede abrir en más para una mejor implementación, por ejemplo con un token para cada palabra reservada, carácter de puntuación u operador.

Consideraciones:

- Debe poner un enum con los tokens en un archivo llamado tokens.h
- La definición de flex no debe tener la sección de código de usuario. El código adicional necesario lo pondrán en el archivo main.c
- El scanner implementado con flex:
  - Devolverá el token en los casos que encuentre un elemento de la gramática.
    - Si el token es identificador o constante mostrará en main el lexema correspondiente
    - Si el token es de solamente un carácter mostrará en main ese carácter (no cadena de un carácter).
  - Ignorará los comentarios
  - Mostrará un mensaje de error para los casos previstos en la implementación.
    - Error léxico común (cadena inválida)
    - Constante inválida
    - Identificador inválido
    - En los tres casos se mostrará el lexema correspondiente
- Debe usar las directivas header-file y outfile.

Sugerencias:

- Utilice las directivas noinput y nounput para evitar warnings innecesarios
- No hace falta reconocer EOF, cuando flex lo lea enviará un token cero, por tanto basta que al armar el enun de los tokens ubique como primero a FDT
- El header generado por flex incluye la declaración de yytext, por tanto y dado que no tenemos necesidad de guardar los distintos lexemas (basta mostrarlos en el momento) puede usar yytext en main para mostrar el lexema

Se incluye el archivo entrada.txt que puede utilizar para hacer pruebas, y el archivo salida.txt que es la salida correspondiente a la entrada de prueba. Por supuesto el formato es solo un ejemplo, no hay necesidad de que lo haga exactamente igual, si que muestre la información que se pide de un modo razonable.

Comandos: En consola, estando parados en el directorio con los archivos del TP y suponiendo que a la especificación de flex la llamó scanner.l debería bastar con: `flex scanner.l` ya que el nombre de archivo de salida debe estar indicado en las directivas.
Suponiendo que generó el archivo scanner.c para compilar puede usar:

```
gcc -g -Wall scanner.c main.c -o scantest -lfl
```

Para ejecutar: `./scantest <entrada.txt >salida.txt`

Entrega: Archivo compactado con los 3 archivos: scanner.l, main.c y tokens.h. En main.c como comentario al principio pongan los datos del grupo y sus integrantes, o sea, la carátula del TP.

Fechas

Fechas de entrega:

K2053 (miércoles): 09/10
K2004 (jueves): 10/10

Fechas de recuperación: es la fecha en que deben haber entregado y correcto.

K2053 (miércoles): 30/10
K2004 (jueves): 31/10