Trabajo Práctico N.º 4

En base a la gramática Sintáctica hecha en el TP2 y la implementación de la gramática léxica hecha en el TP3 armar un programa que usando la herramientas flex y bison reconozca un fuente en lenguaje mini.
En este TP queremos reconocer la sintaxis sin generar el código de seudo ensamblador ni reconocer los posibles errores semánticos.

Consideraciones:

- Tanto en flex como bison deben poner las directivas que eviten tener que agregar opciones adicionales al invocar las herramientas en la línea de comando
- La función main estará en su propio fuente main.c que llamará a yyparse y luego informará si hubo éxito o no y la cantidad de errores sintácticos y léxicos.
- La rutina yyerror debe informar el número de línea en la que ocurrió el error y el error debe ser descriptivo (no solo “syntax error”). Coloque la definición de yyerror en el epílogo de parser.y
- Se debe analizar todo el fuente, no abandonar el análisis al encontrar el primer error.
- Para un mejor seguimiento pondremos en bison acciones semánticas que permitan ver a grandes rasgos que es lo que va analizando, para ello:
  - Luego de leer el identificador del programa mostraremos por pantalla: programa xxx (donde xxx es el nombre del programa).
  - Al final de cada declaración mostraremos por pantalla: entero xxx (donde xxx es el lexema del identificador declaración)
  - Al final de una sentencia leer mostraremos: leer
  - Al final de una sentencia escribir mostraremos: escribir
  - Al final de una sentencia de asignación mostraremos: asignación
  - Al aplicar el operador * mostraremos multiplicación
  - Al aplicar el operador / mostraremos división
  - Al aplicar el operador % mostraremos módulo
  - Al aplicar el operador + mostraremos suma
  - Al aplicar el operador - binario mostraremos resta
  - Al aplicar el operador – unario mostraremos inversión
  - Al cerrar paréntesis mostraremos paréntesis
- ACHATAR GRAMATICA y usar las directivas necesarias para establecer precedencia y asociatividad.

Sugerencias:

- Utilice las directivas noinput y nounput para evitar warnings innecesarios

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.
También se incluyen los archivos entradaerr.txt y entradaerr2.txt con sus correspondientes salidas salidaerr.txt y salidaerr2.txt. Otra vez el formato es indicativo solamente. En particular en los mensajes de error léxico y el lexema que provoca el error.

Entrega: Directorio compactado con los archivos: scanner.l, main.c y parser.y.
Si lo consideran necesario pueden entregar archivos adicionales, por ejemplo más fuentes o encabezados