DUE LAST WEEK OF NORMAL CLASSES

But have it ready a week before for the DEMO

(NOT A PRESENTATION just show what it does)

DEMONSTRATION:

Input file (has "program" code)

Remove extra white space and comments

Format *in an output file*

Output file should be the input to Lex/Yacc process for Syntactic Analysis (CFG)

Output from THIS should be input for Intermediate Code Translation

Output from THIS should be a c++ program

Compile using gcc

Execute

Suppose that the context of text file "final.txt" is:

```
(*  Program Name: Spring 2014
     Final Project
     Name: Nathan Reed
-------------------------------------------------*)
PROGRAM aba13;
VAR

   ab5,  cb  ,  be, eb: INTEGER; (* declare variables *)
BEGIN
   ab5   =  5    ;
   cb  =10;
   PRINT('ab5=',    ab5);
(* compute and print the total of a and c  *)
   eb=  cb + ab5  ;
   PRINT('eb='   ,eb )  ;

   be  -  2*abc5 +  eb;
   PRINT(   be  )   ;
END.
```

Note: Have a program that would go through the text file and *format properly*, outputting in a second text file

1. Remove all documentation, blank lines, and extra spaces from the program. The program should look like the following. You are going to use this program to check the grammar of each statement.

```
PROGRAM aba13;
VAR
ab5, cb, be, eb : INTEGER;
BEGIN
ab5 = 5;
cb = 10;
PRINT(‘ab5=’, ab5);
cb = cb + ab5;
PRINT( cb );
be = 2 * ab5 + eb;
PRINT( be );
END.
```

SPECIFY production rules in flex/bison

2. Given the following CFG (some rules are in EBNF form):

<start> ➜ PROGRAM<pname>; VAR ,<dec-list>; BEGIN <stat-list> END.

<pname> ➜ <id>

<id> ➜ <letter>{<letter>|<digit>}

<dec-list> ➜ <dec> : <type>

<dec> ➜ <id>, <dec> | <id>

<stat-list> ➜ <stat>; | <stat>; <stat-list>

<stat> ➜ <print> | <assign>

<print> ➜ PRINT (<output>)

<output> ➜ [“string”,} <id>

<assign> ➜ <id> = <expr>

<expr> ➜ <term> | <expr> + <term> | <expr> - <term>

<term> ➜ <term> * <factor> | <term> / <factor> | <factor>

<factor> ➜ <id> | <number> | <(<expr>)

<number> ➜ <digit>{<digit>}

<type> ➜ INTEGER

<digit> ➜ 0|1|2|…|9

<letter> ➜ a|b|c|d|e|f

Where PROGRAM, VAR, BEGIN, END. , INTEGER, PRINT are reserved words and all uppercase

flex/bison should print out the errors, but MANUALLY fix the errors, vvv

Your job is to use the given grammar and determine whether the program in part 1 is accepted or not (i.e. whether there are or aren't syntax errors in the program). Your program should produce one of the following error messages as soon as the error is detected:

```
PROGRAM is expected (if PROGRAM is missing or spelled wrong)
VAR     is expected (if VAR is missing or spelled wrong)
BEGIN   is expected (if BEGIN is missing ir spelled wrong)
END.    is expected (if END. is missing or spelled wrong)
UNKNOWN IDENTIFIER if variable is not defined

;       ; is missing
'       ' is missing
.       . is missing
(       ( is missing
)       ) is missing
,       , is missing
=       = is missing
```

(

3. If there are no errors in the program, translate the program into a C++/or another high level programming language. Store the C++ program in “abc13.cpp” file. Your C++ program should look like this:

```
#include <iostream>
Using namespace std;
int main()
{
int ab5, cb, be, eb;
ab5 = 5
cout << ”ab5=” << ab5;
eb = cb + ab5;
cout << eb;
be = 2 * ab5 + eb;
cout << be;
return 0;
}
```

4. If you run the abc13.cpp using the C++ compiler, we should be able to see the correct output