CSC426, Compilers, Fall 2013
YASL Compiler Stage 1: The File Manager

I. Source Code and Environment to Work in
The source code I am providing for this project (see attached) is the only source code I will provide all semester. I will grade your program by running it in addition to looking at your code electronically.

Unlike for most classes I teach, when I grade projects in this course, my primary concern will be functionality (does the program run and perform the required tasks using the algorithms we discussed in class). Issues related to style (comments, modularity, etc.) will not be given direct weight, but will important to you for your own sanity than to me in terms of the grade. **You must add your name and a one sentence comment to the top of any file you create or modify throughout the semester. Other than that, comments are up to you. This is going to be your project… if you make a mess, you will pay the price later in the semester when you need to go back and look at code you wrote several months earlier.**

What to turn in
To turn in the project save the work in a folder named "stage1Lastname" (where Lastname is your last name). Zip the folder up and upload it to Moodle. Also send me an email to tell me that stage1 is ready to grade so I know the final work has been uploaded. The project is due at 5pm on Friday 9/6. Once you send the email you should not edit the project any more.

Do not underestimate the importance of writing neat, well-document, modular code. Also, **do not underestimate** the importance of doing things the way I suggest they be done – if you go off on your own it is possible that you will get a given project to work – but it may not expand nicely into subsequent projects.

A final warning – this project is only worth 2% of the project grade for a reason – it is a very small and very tightly structured project. It is really intended as a 'warm up' for those of you who have been away from C++ for a while. Don't become overconfident – the projects will become successively harder and more challenging.

II. The FileManager
Attached to this document you will find printouts of several programs and a data-file which were discussed in class. You should test these out using the following instructions.

Download the folder stage1vsnet2008 from Moodle. Unzip it. Rename the folder to be stage1YourLastName (where you use your own last name). Start Visual Studio .NET 2008 and select "file, open project" and open the file named yaslc.sln by double clicking on it. In addition to saving the project on your own laptop you can zip it up and upload it to Moodle to save it.

Your goal is to add the following features to the program:

1. Add a new void method to the class fileManagerClass. The method should be named pushBack() and it should not take any parameters. When this method is called, the index into the buffer charToReadNext should be decremented by one. This has the effect of returning the last character to the buffer. If the value of charToReadNext is already zero when this function is called **no changes should take place (i.e. do not decrement the value.)** Make some changes to the driver program so that you can test the function (test this however you like.)

2. Add a void method to the fileManagerClass named printCurrentLine() that takes no parameters. When this method is called the current line number followed by a hyphen a space and the contents of the line (i.e. the contents of the buffer) should be printed. In other words you might print:

    3- Pretty cool, huh!

Do not be cute. Print the output exactly as shown above (nothing more and nothing less.)

3. Add a private *bool* variable to the fileManagerClass named "autoPrintStatus". This variable should be given a default value of *false* in the constructor. Modify the appropriate method so that every time a new line is read from the file it is printed (use the printCurrentLine() function) if the value of autoPrintStatus is *true*. Add a void method named setPrintStatus(newStatus) where newStatus is a bool parameter. This method should change the value of the autoPrintStatus variable to match newStatus. Test your work – this will require that you "rig" the main function so you can test various cases in your code. For example, you might add an if statement inside of the loop that calls setPrintStatus() after reading a specified number of characters. I don't care how the main function ends up looking or what it ends up doing – it is there only to help you test – I will only be grading the filemangr.cpp class.

4. Add an int method named numLinesProcessed() that takes no parameters and returns the cumulative number of lines that have been processed (read into the buffer) so far. Modify the main function so that it prints a final message of the form:

YASLC-XY has just compiled xx lines of code.

XY should be your initials. For example, DB for me, or JS for "John Smith" Just before the program terminates.

5. See notes at the top of this document regarding required adding required comments to the top of each file you modified. Also see notes at the top of this document about turning in your work.