**ECE 1111: Engineering Computation I**

**Homework No. 10: Application Programming – Windowed Grep in C++**

**Deposit your work in:**

**/data/courses/ece\_1111/current/homework/hw\_10/<lastname\_firstname>**

**Goal:** Locating text in a file is one of those extremely fundamental capabilities that Unix excels at with commands such as grep. In this assignment, you will write a simple version of grep that provides some flexibility for how the output is presented. We will use C++ for this assignment.

**Description:** Write a program that has the following user interface:

**mygrep.exe -w “word” -n num\_lines file1.txt file2.txt ...**

The program cycles through all the files specified on the command line and searches each line for a match to *“word”*. The match should be **case-insensitive**. The options “-**w**” and “**-n**” can be specified in any order. The option “-**w**” is mandatory – if it is not specified the program should display an error message and exist. The “**-n**” option is optional with a default value of .

When you find a match, output the lines immediately before and after the match. For example, suppose you specify the options *“***-n 2 -w bOB foo.txt***”* and the file **foo.txt** contains:

**See Jane run.**

**See Mike run.**

**See Mary run.**

**See Bob run.**

**See Alice run.**

**See Tom run.**

**See Barbara run.**

The output should be:

**===**

**File: foo.txt**

**See Mike run.**

**See Mary run.**

**SEE BOB RUN.**

**See Alice run.**

**See Tom run.**

**===**

In other words, you should print the filename that produced the match, line that matched and the preceding two lines and following two lines because *“”*. Print the line that matched in upper case and make sure the match is case-insensitive (note the mixed case in my example).

Do not read the entire file into memory. You can use **fgets** to read the file, but you must use a streaming data approach (meaning buffering the data in a buffer of lines). Think about how you can use an array of character string pointers as a data structure to facilitate maintaining the most recent lines from the file in memory.

Time your code on a set of text files that have one million lines of text in them. Compare the execution time of your code to the Unix version of grep. Put your results in the top of your main program in the comment area.

Implement this program using a driver program that:

* **parses the command line arguments;**
* **instantiates a MyGrep object;**
* **loops over all files specified on the command line;**
* **calls a method named “grep” in your C++ class that uses the following interface:**

**bool MyGrep::grep(char\* filename, char\* word, long num\_lines, FILE\* fp);**

* **outputs match information to stdout by calling grep from your driver program with an argument of stdout.**

You are free to implement any other methods in your class to facilitate this task. You will be evaluated on the organization of your code. Remember methods in C++ should be simple and short.