Name: _____

Problem	Points	Score
1	100	
Total	100	

Notes:

(1) For this exam you are allowed to open a terminal window on your computer, you are allowed to web surf with Google, but you cannot use online chat or other interactive services.

(2) The first step in this exam is to create a workspace in the following directory:

/data/courses/ece_1111/current/exams/ex_03/lastname_firstname/p01

Put all your code here.

(3) Set the permissions using "chmod -R u+rwx,g-rwx,o-rwx <lastname_firstname>" so only you have read and write permission to this directory.

Consider this sequence of Linux commands:

```
ece-000_[1]: echo $PWD
/data/courses/ece_1111/resources/data/text
ece-000_[1]: cat data_v01.txt
Joe joe
Alex alex
Mary mary
ece-000_[1]: cat data_v01.txt | tr ' ' '\12' | sort -f | uniq -ci | sort -nrk1 | awk
'{print $2,$1}' | head -5
Mary 2
Joe 2
Alex 2
ece-000_[1]: cat data_v00.txt | tr ' ' '\12' | sort -f | uniq -ci | sort -nrk1 | awk
'{print $2,$1}' | head -5
THE 13107
OF 6182
AND 5772
TO 4269
A 4149
```

Write a C++ program that does exactly the same thing and produces the same result for the files `data_v0[0,1].txt` in `/data/courses/ece_1111/resources/data/text.` The only difference is your program need not sort the output into descending order. The output can be unsorted.

Obviously, work with `data_v01.txt` first and make sure your code produces the correct output.

To make things easy, you can assume a word is defined as any set of characters between two spaces, a space and a newline character, or any combination of the two. You don't need to worry about edge cases like hyphenation, numbers, non-alphabetic characters, etc.

Your program should take its filename from the command line:

```
p01.exe <filename>
```

You can assume only one file is input.

You must implement this using four files: Makefile, header file (`p01.h`), driver program (`p01.cc`), implementation file (`p01_00.cc`). All your class methods must be implemented in `p01_00.cc`.

Create a class called MyHistogram that has the following methods:

```
MyHistogram::load_words(???, char* file);
MyHistogram::count_words(???, ???);
MyHistogram::print(???, FILE* fp);
```

The method `load_words()` takes a filename as an input and returns some sort of data structure that contains a list of words that occur in the file. The nature of the data structure is up to you. But you must pass this data back to the driver program in a way that is safe (e.g., do not pass pointers to class data back to the driver program; create copies of the data). Convert the words to uppercase when you read them, so the histogram is case-insensitive.

The method `count_words()` takes your data structure as input and returns a list of unique words and the number of times they occur. Again, you are free to define whatever data structure you want to hold this data. You **DO NOT** need to sort these by frequency of occurrence, and your code does not need to be efficient. We will focus on results for small test cases.

The method `print()` takes your data consisting of the unique words and counts and prints the UNSORTED histogram to the file pointer `fp`. Your driver program should set `fp` to `stdout`.

Your program should work for any text file of any length. I will test it on data similar to `data_v00.txt`, but similar in length to `data_v01.txt`.

Make sure there are no memory leaks in your code, which means you are most likely going to need a destructor method.

Start by developing the `load_words` function. Successful completion of this function gets you a 75 on this exam. Completion of `count_words` raises your grade to 85. Completion of `print` raises your grade to 90. A well-written class with no memory leaks that is functionally correct gets you a 100 and three gold stars!