Name: _____

Problem	Points	Score
1	50	
2	60	
3	70	
4	80	
5	90	
6	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

(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. Create six subdirectories within this directory: p01 through p06. Put ALL your code in these directories. Do not touch your files after the exam is over.

(4) After you complete p01, copy the code into p02 and edit it. You do not need to change any filenames.

(5) You must use a make file, a header file and a main program. You must use a class definition, or you will get 0 points for this exam.

This exam is structured similar to the way the final exam will be structured. Start with problem 1. Put the code in a directory p01. Call the main program and its components example.*. When done, copy your code to the next problem (e.g., p02) and continue editing it. Only turn in code that is completely working. I will only grade the highest level you submit. Delete any directories above the one you want graded.

If the level you submit for grading doesn't work, you will get a 50 for this exam (we call this the "you didn't debug your code and wasted my time" penalty). Therefore, what you submit must work and meet the stated requirements.

Problem No. 1: Create a simple class called Exam that consists of a default constructor with no arguments, a default destructor and a method called myprint that takes a file pointer as an argument and prints "exam no. 3". Put the implementations of your functions in example_00.cc, not in the header file. Your main program should construct an object ("Exam exam;") and call its print method ("exam.myprint(stdout)"). It should accept an argument from the command line using argv[1].

Problem No. 2: Add a class method called myread that accepts a filename as an argument ("myread(char* fname)"). Have it open a text file, read it line by line and print the contents of each line to stdout. It should close the file. It should not have any memory leaks. Add a line to your main program that calls this method using argv[1] after you construct an object ("myprog.exe myfile.txt" is how I will run this).

Problem No. 3: Add a class method called mylen that computes the length of a character string. It accepts a character string as input and returns the length of the string as an integer. Add a line to your myread function that calls this method for each line in the file and prints the length of the line. Note that you must strip the trailing newline character.

Problem No. 4: Copy the name of the file that you opened to class internal data. Do not copy the pointer – copy the data. Make sure you delete any memory allocated in the destructor. Change your myprint() function to print the filename being processed ("filename: %s") to the file pointer passed as an argument.

Problem No. 5: Change myread() so that it stores all lines in the file as class internal data. Modify myprint() so that it prints this data to standard out. Add a line to your main program that calls myprint() after you call myread(). Myprint() is the only function that should print the lines to stdout; remove the print statements from myread(). Make sure your destructor cleans up memory. Change your main program to use new to construct the object and delete it at the end of the program:

```
Exam* exam = new Exam;
…
delete exam;
```

Put a print statement after delete that prints "hello world" just to make sure your program doesn't crash in the delete method.

At this point, your program reads a file line by line into memory and prints its contents to stdout.

Problem No. 6: Add a method called compute that has no arguments and sums up the lengths of the lines in the file stored as class data. This method returns an integer which is the sum of the number of characters in all the lines. Add a line to your main program that calls this method and prints the return value using a %d format:

```
the total number of characters = <sum>
```

"<sum>" is replaced by the actual sum.