Name: ____

Problem	Points	Score
1	75	
2	25	
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 two subdirectories within this directory: p01 and p02. Put ALL your code in these directories. Do not touch your files after the exam is over.

(4) You must use a make file, a header file and a main program file named p01.cc (or p02.cc). All other code needs to go into an implementation file called p01_00.cc (or p02_00.cc).

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

```
/data/courses/ece_1111/current/exams/exam_03/lastname_firstname
```

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 two subdirectories within this directory: p01 and p02. You will use these for problems 1 and 2 respectively. Put ALL your code in these directories. Do not touch your files after the exam is over.

You must use a make file, a header file and a main program file named p01.cc (or p02.cc). All other code needs to go into an implementation file called p01_00.cc (or p02_00.cc). Failure to follow these instructions will result in a grade of 0.

Problem 1: Create a simple class called MyLongVector that has:

- protected data, declared in this order: (1) a long integer, ndim_d, containing the number of elements in the vector, and (2) a long integer pointer, data_d, that holds an array of integers;
- a default constructor (no arguments), that initializes the dimension to -1 and data_d to NULL;
- a default destructor that cleans up memory (see the description below);
- a public member function, load, that reads a file (see the description below) and sets the internal data;
- a public member function, myprint with a file pointer argument, that prints the values of all internal data;
- a private member function called "cleanup" that deallocates memory.

Create a driver program, p01.cc, that reads integers from a file and assigns them to this class. For example, let the file "foo.txt" contain the following data:

```
nedc_999_[1]: cat foo.txt
1
2
3
4
```

Your program should read this file, store the data in the internal array, and print it:

```
nedc_999_[1]: p01.exe foo.txt
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
```

Your program must work for any size file and not have any memory leaks. Note the output above does not include the debugging information requested below.

Your driver program should consist of the following statements:

```
MyLongVector* v1 = new MyLongVector;
MyLongVector* v2 = new MyLongVector;
v1->load(argv[1]);
v2->load(argv[1]);
v2->myprint(stdout);
delete v1;
delete v2;
```

The load method does three things: (1) calls the method "cleanup" which functions as described below; (2) reads the file once, counting the number of lines, and creates a vector of length equal to the number of lines in the file; (3) reads the file a second time and assigns values in the file to the internal data vector, data_d. Note that to do (3) you will need to allocate memory in the load method. You can add this code to load() directly or you can create an additional method called "allocate" to do memory allocation (preferred). That is up to you. If you add an allocate method, its argument should be the new dimension of the vector.

The method "cleanup" prints the message "…cleaning up memory…" and checks if the dimension is -1. If so, it returns. Otherwise, it deallocates any allocated memory that has been previously allocated, sets the dimension to -1, and sets data_d equal to (long*)NULL.

The destructor does two things: (1) prints the message "…destroying the object…" and (2) calls the cleanup method. If everything works properly, your program should have no memory leaks.

The constructor prints the message "…constructing the object…". The load method prints the message "…loading the object from a file…".

Problem 2: Create a simple program that creates a vector of vectors using the class in problem no. 1:

```
MyVectorLong* foo = new MyVectorLong[2];
```

Show where in memory each data element is stored for the objects foo[0] and foo[1] and how much memory each object takes. Do this using print statements.

How much memory is consumed by the vector foo? Is it two times the size in memory of foo[0], or are there other considerations.

Comment on what you observe and the implications of your observations. Be as detailed as possible in describing where memory is allocated.

Put your memory analysis results in a file AAREADME.txt in your p02 directory.