Name:

Please remember you must follow instructions exactly in this course. Submit your code in the usual place on the class server – *qu\_08/lastname\_firstname/*.Your main program must be called *p01.cc*. Your functions should be in a file *p01\_00.cc*. Your header file should be called *p01.h*. Your binary must be called *p01.exe*. You must use a make file – the command “make” generates *p01.exe*.

The requirements for this program are:

Write a C program that does the following:

1. Create a program, p01.cc, that takes an integer, , as an argument.
2. Create a structure that holds a float, a long and a character pointer. Call it mystruct. You can typedef it to MYSTRUCT.
3. Create a function, called myalloc, that allocates an array of structs inside the function and returns a pointer to the main program. The function should take an integer as an argument that represents the number of structures to be allocated (the length of the array of structs). You must allocate the array of structs inside the function – not in the main program.

Inside this function, set the float to a random value using the math library function drand48(). Set the long to an integer value indicating the position in the array (e.g., the first element has a value of zero, the second element as a value of 1, …). Set the char\* pointer to hold a string **“ece\_1111”** (same string for all elements of the array, but separate memory spaces). Be sure to allocate memory for this.

1. Create a function called myprint that takes three arguments:

**myprint(FILE\* fp, long N, MYSTRUCT\* array);**

and prints the values of all the elements of array *\*and\* they memory locations they reside at*. represents the number of structures (elements) in the array, and can be an open file or stdout. From your main program, call this function using stdout for .

Your main program should go through these steps:

1. Decode the command line.
2. Declare an array of structs: MYSTRUCT\* array;
3. Call myalloc().
4. Call myprint using .
5. Exit.

Your output should look something like this:

ece\_000[1]: p01.exe

struct No. 1:

float: memory location = 1234567890, value = 1.2895

long: memory location = 1234567894, value = 1

char\*: memory location = 1234567902, value = ece\_1111

struct No. 2:

. . .

Note this program has a serious memory leak, but we will deal with that later (e.g., exam no. 2).