**ECE 1111: Engineering Computation I**

**Laboratory No. 11: Function Pointers and Macros**

**Deposit your work in:**

**/data/courses/ece\_1111/current/labs/lab\_11/<lastname\_firstname>/[p01, p02, p03]**

**Goals:** Function pointers and macros are two tools that allow us to write simple yet general code that can be used with all data types. In this lab, you will learn how to use library functions that accept a function pointer (e.g., qsort) and how to write a macro that can work for any data type.

**Deliverables:**

1. *Source Code:* a collection of files in one directory that meets the assignment’s requirements (e.g., Makefile, header file, main driver program).
2. *Check Off:* Demonstrate that your program runs successfully. Answer questions about data types and scope.

**Description:**

1. (/p01) Create a C program that uses the C library function qsort (or its variants) to sort a file containing a list of numbers. Create a text file that contains 20 random integers between [-100,100]. Make sure no integer appears twice and there are an equal number of odds and evens, and negative and positive numbers. Sort this file using the C library function qsort using the following comparison function:

**if A is even and B is even:**

**do a normal numeric comparison:**

**if (A < B), return -1,
if (A = B), return 0,
if (A > B), return 1.**

**if A is odd and B is odd:**

**reverse this comparison (e.g., A < B returns 1).**

**if A is odd and B is even, or A is even and B is odd:**

**if (abs(A) < abs(B)), return -1,
if (abs(A) = abs(B)), return 0,
if (abs(A) > abs(B)), return 1.**

Your main program should have this interface:

**p01.exe input\_file.txt sorted\_file.txt**

The first argument is the input file, and the second argument is the output file (one integer per line);

**ece-000\_[1]: cat input\_file.txt**

**27**

**-3**

**0**

**...**

The output file should be in the same format. You should construct a test case for the example above manually and use it to debug your code.

Your program should work for a file of any length.

Test your program on small cases that you can work manually before you try to run the entire test case. For the final test case, generate 100,000 random numbers and sort them. Time your program.

1. (/p02) Repeat No. 1, but instead of operating on two long ints, create a structure called BigNum that has one data member, a long int named “number”. Modify your program to create and sort structures using the same comparison function as in No. 1. Show that for the same number sequence you used in No. 1, you get the same result (the structures are sorted in the same order). I’ll leave it to you how to convince your TA that your program is producing the same result as No. 1 – but your demonstration must be convincing!
2. (/p03) Write a program that loops N times, generates two random integers in the range [-1010,1010], and compares them using the comparison function in no. 1. However, implement this function as a macro. Demonstrate that your function works for both long integers and floats without any source code modifications. Your main program should have this interface:

**p03.exe 999**

where the number of iterations, N, is argv[1]. For each iteration, print the iteration number, the two input values, and the value returned by your comparison function. Again, I’ll leave it up to you how you convince the TAs that your program is working, but your demonstration must be convincing.

**Summary:**

Function pointers and macros allow us to create code that is efficient, general and yet minimizes complexity. Many advanced programming languages like Python rely on these mechanisms.