# ECE 71 – Introduction to Computational Programming

Professor Kriehn – Fall 2017

**Code Due By:** Midnight on Thursday, October 19, 2017
**Write-up Due By:** Class on Mon/Tue, October 23/24, 2017

## HOMEWORK #20 – Number Sorting with Files

Write a program that merges numbers from two files and writes all the numbers into a third file in ascending order. Each input file contains a list of 50 sorted double floating-point numbers from the smallest to largest. After the program is run, the output file will contain all 100 numbers between in the two input files, also sorted from smallest to largest. Format the output into two columns – the first column contains the numbers 1-100, and the second column contains the appropriate number from the text file. Also print the results to the screen.

**Specifications:**
The input file names are called "**`numbers1.txt`**" and "**`numbers2.txt`**", and can be found on the website for ECE 71. Copy and paste the numbers from each text file into a "`numbers1.txt`" and "`numbers2.txt`" within your project directory. This can be done by creating a project with a `main.cpp` source file, compiling it, launching a terminal window, and using `nano` to create a text file where the numbers can be copied into (separately for each file). Once the numbers are copied into a generic text file, save it with the appropriate filename and exit `nano`.

The program should use two functions with the following function prototypes:

```
void checkFailure(ifstream &fin1, ifstream &fin2, ofstream &fout);
void sortNumbers(ifstream &fin1, ifstream &fin2, ofstream &fout);
```

The `checkFailure()` function should accept the two input-file streams and the output-file stream as call-by-reference arguments. The function should make sure that the files were opened properly and exit the program with `EXIT_FAILURE` if necessary.

If a file fails to open, the error should read:

```
Input file "[filename]" failed to open.
```

`[filename]` represents the file that failed to open. For instance, if `numbers2.txt` fails to open, the error message to be printed to the screen is:

```
Input file "numbers2.txt" failed to open.
```

The `sortNumbers()` function should also accept the two input-file streams and the output-file stream as call-by-reference arguments. The function should read in the values from the input files (without using an array) and compare values one at a time to determine the smallest value between any pair of numbers. As a new number is read in, the smallest value should be written to the output file.

The first column written to the output file should be an integer between 1-100, representing which value is being represented, since there are a 100 numbers total. The integer values 1 - 100 should be

printed with a width of 3 and right justified.

The second column written to the output file should be the appropriate double floating-point number. Print each number with a precision of 8 values after the decimal point. The two columns should be separated by a tab.

The final output should print the results to the screen, and to a file called "**`sorted.txt`**".

**NOTE:** When you first read in the first two numbers, one of them is the absolute minimum, and is the first number written to the file. If that number came from `numbers1.txt`, then the next number to be read in should also be from `numbers1.txt`. If the new number read in is still smaller than the other number from `numbers2.txt`, write it to the file and repeat the process until the next number read in from `numbers1.txt` is no longer the smallest number. Then write the number from `numbers2.txt` to the file and read in a new value from `numbers2.txt`. Repeat the process as necessary (i.e., read in a new number one at a time from the appropriate data file) to ensure each value is written to the data file in the correct order. If/When you run out of numbers from one of the input data files, any remaining numbers in the second input data file should be read in and written to the output file directly, since comparisons are no longer necessary.

As an example, if you execute the program with the following underlined inputs, the output will be:

```
~> main.o
  1       0.00476587
  2       0.00952649
  3       0.01107230
  4       0.01862460
  5       0.02598610

  ...

 99       0.98603100
100       0.99549800
~> more sorted.txt
  1       0.00476587
  2       0.00952649
  3       0.01107230
  4       0.01862460
  5       0.02598610

  ...

 99       0.98603100
100       0.99549800
~>
```

Develop your I/O diagram and pseudocode, debug your code, and submit to the Grader Program.