ECE 71 – Introduction to Computational Programming

Professor Kriehn – Fall 2017

Code Due By: Midnight on Friday, October 27, 2017
Write-up Due By: Class on Mon/Tue, October 30/31, 2017

HOMEWORK #23 – Statistical Functions

Write a program that prompts the user to enter a set of N double precision floating point numbers, scans them into a 1-D array, and calculates various statistical properties of the array of numbers, including the maximum, minimum, average, variance, standard deviation, standard error, and median.

Below are the mathematical definitions of interest. If you have an array $x[]$ with N elements:

Maximum – The maximum finds the maximum of the array elements *x[i]*, where *i* ranges from *0* to *N-1*.

Minimum – The minimum finds the minimum of the array elements *x[i]*, where *i* ranges from *0* to *N-1*.

Average – To find the average $\bar{x}$, sum the array elements $x[i]$ and divide by the number of elements N:

$$\bar{x} = \sum_{i=0}^{N-1} x[i] \Big/ N$$

Variance – To find the population variance σ^2, sum the square of the difference between each array element $x[i]$ with the average $\bar{x}$, and divide by the number of elements N.

$$\sigma^2 = \sum_{i=0}^{N-1} (x[i] - \bar{x})^2 \Big/ N$$

Standard Deviation – To find the population standard deviation σ, take the square root of the variance:

$$\sigma = \sqrt{\sigma^2}$$

Standard Error – To find the standard error, divide the standard deviation by the square root of the number of elements in the array N:

$$SE = \frac{\sigma}{\sqrt{N}}$$

Median – The median finds the middle value of an array of elements. For an odd number of elements, the median is the central array element, after the array has been numerically sorted.

For an even number of elements, the median is the average of the two central elements, after the array has been numerically sorted.

NOTE: For this function to work, you will also need a *sort()* function that sorts the elements in ascending order.

Specifications:
Use the following function prototypes:

```
double maximum(const double a[], int n);
double minimum(const double a[], int n);
double average(const double a[], int n);
double variance(const double a[], int n);
double stdDev(const double a[], int n);
double stdErr(const double a[], int n);
double median(double a[], int n);
void sort(double a[], int n);
```

Assume an array size of 100 elements. Prompt the user to enter the number of elements to be used (up to 100) and use that to store the numbers in a subset of the array. Use a `for` loop to prompt the user to enter in a number for each element, and read in the appropriate numbers. Your program should then call the appropriate functions and print the results to the screen using 2 values after the decimal point. If you execute the program, the following information should be displayed:

```
~> main.o
Welcome to the Statistical Functions Program
Number of values to be entered: 300

Only up to 100 values can be entered.
Number of values to be entered: 3

Enter number 1: 3
Enter number 2: -1
Enter number 3: 4

The maximum is: 4.00
The minimum is: -1.00
The average is: 2.00
The variance is: 4.67
The standard deviation is: 2.16
The standard error is: 1.25
The median is 3.00

~> main.o
Welcome to the Statistical Functions Program
Number of elements in the array: 4

Enter number 1: -5.2
Enter number 2: 2.6
```

```
Enter number 3: 6.3
Enter number 4: 19.3

The maximum is: 19.30
The minimum is: -5.20
The average is: 5.75
The variance is: 78.43
The standard deviation is: 8.86
The standard error is: 4.43
The median is: 4.45
```

HINTS: For the maximum and minimum, initially set the maximum/minimum to the 0^{th} element of the array. Then compare that value against the other elements and store find the true maximum/minimum. Use the math library and the `pow()` function for the variance. Use the math library and the `sqrt()` function for the standard deviation and the standard error. The median function must call the sort function.

HOMEWORK #24 – Number Sorting with Files

Write a program that merges numbers from two files and prints the results in ascending order. Each input file contains a list of 50 sorted double floating-point numbers from the 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 files. Use an array and a sort function to solve the problem.

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 four functions with the following function prototypes:

```
void checkFailure(ifstream &fin1, ifstream &fin2);
void inputData(ifstream &fin1, ifstream &fin2, double a[], int n);
void sortData(double a[], int n);
void printResults(const double a[], int n);
```

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 `inputData()` function should read in numbers from the 2 data files and store the results in the 100 element array. The values can be read into the array in any order.

The `sort()` function should sort the numbers in the array, as appropriate.

The `printResults()` function should print the sorted numbers using two columns. The first column printed to the screen should be an integer between 1-100, indicating 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.

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
```

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