Standard Deviation

Like the bits assignment, this assignment borrows from one of Dr. Jaffe's assignments. It differs mainly in that this one uses doubles and is broken into functions. Breaking the problem into simple functions and keeping it simple by using functions, is an important part of the experience.

Write a program that declares an array of 100 doubles and then fills it with 75 random doubles between 40.0 and 60.0 (from 40.0 up to be not including 60.0) in value. Print out the values in the array then print out the largest value, the smallest value, the average value and the standard deviation of the set of 75 values. Assuming you run on prclab, without calling srand(), here's what your printout should look like:

```
number[0] is 56.804
number[1] is 47.888
number[2] is 55.662
number[3] is 55.969
number[4] is 58.233
number[5] is 43.951
number[6] is 46.704
number[7] is 55.365
number[8] is 45.555
number[9] is 51.079
number[10] is 49.548
number[11] is 52.577
number[12] is 47.296
```

... and so on

```
The sum of the 75 values was 3784.375.
The smallest was 40.326, and the largest was 59.978.
The average of the values was 50.458.
The population standard deviation was 5.871.
```

Notes:

1. There is no complexity in this assignment! So don't panic and start writing crazy code. Do this assignment as a series of steps, in the same order as the list of function prototypes.
2. Each function is limited to doing one simple thing, and doing it well. Several of the functions call another functions for part of the work that has to be done. Main does not pass anything other than the array and the length of the data. YOU MAY NOT CHANGE A FUNCTION PROTOTYPE.
3. Don't use the `randomBetween()` from Assignment 4. Instead, to back and reread the UsingRandInC.pdf file located near the bottom of the course home page in Canvas. This time you want to implement it for doubles, using the formula involving limits instead of bounds.

4. You already did min, max, sum, and mean calculations in Assignment 2.
5. To compute the standard deviation, you'll need to take the square root of the variance. Use another function to calculate the variance. In the function to calculate the variance, you need to know the mean. Use another function to calculate the mean. Then compute the deviations at each value and square that. To square the deviation values for the variance, multiply the deviation value by itself. The cleanest way to do that is to use a variable to hold the difference and then multiply that by itself. The variance is itself a kind of sum, so add each new deviation-squared value to an accumulating sum of squared-deviation values, the same way you do for computing the sum of simple values. In the standardDeviation function, use the `sqrt` function from the math library (`#include <math.h>`). In addition to including `math.h`, you may need to give the compiler an extra flag to incorporate the math stuff (don't ask me why). The compiler command for this assignment will be `gcc -lm ...` (The `-l` option [that's a lower case letter '`l`' there] provides the compiler with information about a library you want searched for your `#include` files, `-lm` tells it to search the math library. I have no idea why that's not on the default list of places to be searched, although I'm sure there's a good reason.)

   Here is another website that explains how to compute a standard deviation. Don't make it out to be more than a few simple steps: https://www.sciencebuddies.org/science-fair-projects/science-fair/variance-and-standard-deviation.

6. **Important Note:** There is a little detail about sample variance vs. population variance, which boils down to n vs. n – 1. What am I talking about? Like all questions, you can simply Google it: "sample vs. population standard deviation". Use population variance, since that is a tiny bit simpler. Much more important: now that you know that there is that issue, you should include a comment that explains which one you are computing. However, I included it in the name. That way you don't need a separate comment.
7. This problem, like most real world programs, will benefit from "build-a-little, test-a-little". Start with two simple arrays for testing, and call the function you are working on from main as you are working, to test it out:

   ```
   double test1[4] = {1.0, 1.0, 1.0, 1.0};
   double test2[3] = {1.0, 2.0, 3.0};
   ```

   1. Call randomBetween() from main a few times (from in a printf statement) tp make sure it is working and returning values between 40 and 60.
   2. Declare your array, fill it with random doubles, and print them all out to make sure you're using `rand()` properly and they are between 40 and 60.
   3. Now leave the random array stuff alone, and focus on using your 2 simple test arrays for the functions doing stats.
   4. Call the minOfArrayValues from inside printf statements.
   5. Call the maxOfArrayValues from inside printf statements.
   6. Call the sumOfArrayValues from inside printf statements.
   7. Call the meanOfArrayValues from inside printf statements.

8. Call the populationVariance, but start by giving a length of 1. When that value is correct (if should be 0 since there is no difference between that 1 value and its mean), give it a length of 2

```
/* %%% TODO: File header goes here %%% */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* use rand() to get a random double in given range */
double randomBetween(double lowerLimit, double upperLimit);
/* assign random doubles to array elements */
int fillRandom(double *array, int numberToFill,
               double lowerLimit, double upperLimit);
/* print the array values one above the other */
void listTheArrayValues(double* array, int arrayLength);
/* return the minimum value found in the array */
double minOfArrayValues(double* array, int arrayLength);
/* return the maximum value found in the array */
double maxOfArrayValues(double* array, int arrayLength);
/* return the sum of the values in the array */
double sumOfArrayValues(double* array, int arrayLength);
/* return the mean of the array, using sumOfArrayValues() to get the sum */
double meanOfArrayValues(double* array, int arrayLength);
/* return variance of the array, use meanOfArrayValues() to get the mean */
double populationVariance(double* array, int arrayLength);
/* return standard deviation, using sqrt() and varianceOfArrayValues */
double populationStandardDeviation(double* array, int arrayLength);

#define ARRAY_SIZE 100
#define DATA_LENGTH 75
int main(void) {
    double myArray[ARRAY_SIZE];
    int lengthFilled = fillRandom(myArray, DATA_LENGTH, 40.0, 60.0);
    listTheArrayValues(myArray, lengthFilled);
    printf("The sum of the %d values was %.3f\n", lengthFilled,
           sumOfArrayValues(myArray, lengthFilled));
    printf("The smallest was %.3f, and the largest was %.3f\n",
           minOfArrayValues(myArray, lengthFilled),
           maxOfArrayValues(myArray, lengthFilled));
    printf("The average of the values was %.3f\n",
           meanOfArrayValues(myArray, lengthFilled));
    printf("The population standard deviation was %.3f\n",
           populationStandardDeviation(myArray, lengthFilled));
    return 0;
}
```