CISC 3110

Lab 2: Fun with Pointers

In this lab, you'll work with a few programs to get some practice writing and revising pointer-related code. Download and unzip http://bc-cisc3110-s17.github.io/lab2/lab2.zip.

Part 0: Pointer Problems

When you unzip the lab2 file, you'll see two subdirectories. Start with program 0. There's an executable waiting for you, called minmax. Run it. It probably won't work. So, your first task is to fix it. (You might want to use the line of code that's marked 'for debugging.')

Part 1: Converting to Dynamic Allocation

Now consider the code in `program1`—a simple program that reads in numbers, then calculates their mean and standard deviation. Modify the code so that the user first specifies how many input values are provided, then reads them and performs the calculation. Thus, the initial output of

```
Please enter 10 numbers.
```

Should be preceded with a prompt

```
How many input values are there?
```

Once the user answers this question, then the next prompt should be

```
Please enter [?] numbers
```

where [?] is the user's response to the how-many prompt.

Which elements of the program need to be changed? Make as few changes as possible; then (of course) make sure the modified program compiles and runs correctly.

Is there any possibility of memory leak in this new version of the code? If not, why is it "safe"? If so, which element of the program is responsible for preventing memory leaks?

Part 2: Adding more Dynamic Allocation

Now, let's add a little more behavior to the `do_stats()` function. In addition to calculating the average and standard deviation, we want it to return an array that reports the distance of each input value from the mean. That is, if the input is the array {2.0, 3.0, 10.0} (with a mean of 5.0), then the returned array should contain {3.0, 2.0, 5.0}. Note that distances are always non-negative.

Again, change as few elements as possible, and make sure the program compiles and runs correctly. In addition to printing the mean and standard deviation, your main program should print the returned array of distances.

Is there any possibility of memory leak in this new version of the code? If not, why is it "safe"? If so, which element of the program is responsible for preventing memory leaks?