Andrew Clos
CS-161, Section 400
Homework 3a – Assignment Reflection

Understanding:

From the outset of the project I had a pretty good idea that I'd use a *for* loop since the user would be specifying the number of integers to analyze up front. Within the *for* loop, I would plan to have two *if* statements to determine whether each integer was larger or smaller (or equal) to the previously stored values of min and max.

I learned the importance of having sound logic and rigorous test cases while I was testing during coding of the program. I wanted to write the code as efficiently as possible but be "safe" in terms of making sure there were no logical errors or unexpected results. In a trial to make the code more efficient, I tried to use an if and else statement, but learned quickly that I for each iteration of the loop, I need to be able to update either the min or max variable **or do nothing.** If I didn't leave a way out (neither variable gets updated), I could potentially have erroneous results (see code example below in the Test Plan section)

Testing Plan:

I faced two main hurdles during the testing phase of writing this program.

First issue: I initialized the minInt and maxInt variables to zero, but this caused unexpected behavior when the number of integers the user entered was only one or if all the values were either above or below zero.

Second issue: Tied to the first issue, but after the user entered their desired number of integers, the *for* loop began. Because all the data entry was occurring within the *for* loop, the program would not always have the correct values for min and max stored.

I solved these two problems by realizing that the number of integers the user entered would always be 1 or more. Therefore, I could take the first integer entered before the loop, initialize the min and max to this variable and then move to the loop if there were more integers to enter.

My testing plan should have initially included more cases to more rigorously test the code. For example, alternating high and lower numbers, very large or very small numbers. During testing, I wanted to see if an if/else combination would work instead of two *if* statements. My logic was proven to be flawed though if the entries alternated between larger and smaller numbers. I learned that I could fix the situation with an if and else statement instead, or just keep the two if statements.

For example:

Flawed logic, resulted in an integer entry sequence of 1, 5, 3, 1 displaying min: 1, max 3:

```
if(currentInt <= minInt) {
   minInt = currentInt;
}
   else
      maxInt = currentInt;
```

Andrew Clos
CS-161, Section 400
Homework 3a – Assignment Reflection

Working Logic as submitted in project 3a:

```
if(currentInt < minInt) {
   minInt = currentInt;
}
if(currentInt > maxInt) {
   maxInt = currentInt;
}
```

Design:

One item I didn't anticipate when I wrote the project plan, was that I'd need to take in the user's first integer outside of the *for* loop and initialize both of the min and max variables to this value before starting the loop in order to have correct values stored in both the min and max variables in cases of only one integer being entered, or all user entered values being high or lower than the default initialization value. These two concepts of initializing variables to a user defined value and having the program receive the first entry (of potentially 1 or more) before beginning the loop are something I will remember and use going forward.

I also learned from the Project Plan feedback, that I should be more detailed in writing pseudo code (discuss the declaration/initialization of variables). Finally, I wrote the program after submitting the Project Plan, but learned several things during the actual writing of the program that I hadn't discovered while writing the plan. I didn't realize the need to receive the first integer and initialize the min and max variables to this value outside of the *for* loop until I was coding. I should have revised and resubmitted the project plan with these additional details.

Implementation:

Nearly all the problems I encountered were during implementation, which probably says that my Project Plan pseudocode and test cases needed to be more detailed and rigorous. The problems I encountered and had to correct were:

1. The loop needs to be able to update the min or max variable or do nothing. At one point I left out the ability of the loop to not update any value and caught this by entering the right sequence of test input (that should have been part of the test plan).
2. That the first integer would need to be entered before the loop and that I'd need to initialize the min and max variables to this integer before starting the *for* loop. I was proud to have figured this out myself while coding the program, but did check the Piazza forum to see if I was on the right track and noticed a post from our instructor suggesting this was a good strategy to take. This idea was also confirmed when I received feedback from my Project Plan.

Improvement:

Andrew Clos
CS-161, Section 400
Homework 3a – Assignment Reflection

Develop rigorous test cases and implement them methodically! I almost submitted a program that would have produced unexpected results if the sequence of entries alternated between higher and lower values (if / else example above). If I had a more thorough and exhaustive test procedure, I would have been sure to catch this. In the future, writing a script to include all of these test cases and rapidly run through iterations of the program would probably be a good idea and I hope to learn this technique here or in a future course.

I relied too heavily on the example pseudocode and test plan given in the Project Plan, but it was not thorough, and I learned that I need to be more in depth when writing pseudocode and test plans in future assignments.

I also think may be a good idea for me to revise the Project Plan as I begin to write pieces of code for the program, since I seem to make many learning discoveries and form more cohesive logic while I am coding. I would imagine that I will be well served by breaking larger projects in to chunks, that each include their own mini-project plan.