CSCI-1200 Data Structures — Spring 2016
Extra Credit Homework — Debugging

Read this whole assignment before you download or start working on the code. This homework's format is different from the others, so be sure you have read and understood *all* of the instructions.

In this assignment you are given some legacy source code for an existing, complete project that decrypts a text file. The program will run a suite of tests and operations on given data, which eventually decrypts a text file. While the whole program exists, it is in need of some serious debugging.

You must submit your version of the program with fixes for bugs that you have found, but that is only one part of the homework. You are required submit a detailed writeup of what you did to debug the program. **You must submit a PRINTED copy of this during lab on Wednesday May 11, 2016.** See below for more details on the submission.

Your writeup should be thorough. Take lots of notes and some screenshots during your debugging process, wherever you think one would contribute to the text. Every bug you find should be fixed in the code and documented in your report, including why the fix is good. Please do not screenshot every bug, though; 10 to 15 of them should be sufficient.

Use any method of debugging you like, as long as you clearly describe the tools and process you used.

Note on Academic Integrity

Normally, talking about the solutions to problems with other students is fine as long as you do not share code. However, the whole point of this homework is about learning how to find, diagnose, and fix bugs in a program. **Discussing bugs with other students is thus an academic integrity violation.**

Grading Scheme

This assignment is worth 20 points towards your homework grade. There is *no penalty* for not submitting this assignment, it is entirely optional. You will not be able to earn more than 100% on the homework portion of your final grade.

While this assignment is the size of a standard Data Structures homework, you will only be able to earn 20 points maximum. You will earn one point per bug you have found and written up. While there are more than 20 bugs, you cannot earn more than 20 points.

Late Days

As this assignment is due on the last day of classes, **you will NOT be able to use any late days on this assignment**.

Details on the Program

The code will run a number of (buggy) functions that each perform operations on different types of data. Each one is designed to return a specific value; however, the bugs in them will cause them to crash or return the wrong value. By finding and fixing all the bugs, you can guarantee it to return the correct answer.

For example, consider the first function, `arithmetic_operations`. It starts by initializing 19 different variables with various mathematical operations, and its comments indicate what values they should hold to pass all the tests and return the correct value. However, it is fairly easy to notice that these variables are not being calculated correctly, since the program will crash or fail an assert on one of its own tests. Thus, the "bug fix" here is simple: make sure the variables get the correct values.

Remember, there are more bugs to fix than points that you can earn. You will only be given up to 20 points of credit on this assignment. When all bugs are successfully fixed and the program is run and passed in the encrypted file as an argument, it will decrypt and print the contents of the file. It will be obvious when it has been decrypted correctly — you will know when you have it. If you get this far, please include the decrypted text in your writeup.

Downloading and Running

Please download the 5 C++ code files for the decryption program and the one encrypted text file, called `encrypted-goal.txt`. Before you begin debugging, please read the comments especially those at the beginning of each file. Note that the `main.cpp` file contains NO bugs.

The program should be compiled with the following command. The `-lm` flag tells g++ to explicitly include the math library, which otherwise might not get linked correctly on some platforms.

```
g++ main.cpp operations.cpp triangle.cpp -lm -o main.exe
```

If you cannot run this command and generate an executable, see a mentor or TA. **The program should compile when you first download it.**

The program can be run with the following command, taking the encrypted goal as its one argument:

```
./main.exe encrypted-goal.txt
```

The Writeup

There is no exact format to the writeup. Here is what we are looking for:

1. Readability: Please be concise and use complete sentences.
2. Clarity: A few concise sentences per bug that describe why it is causing the program to fail and how you found it. For non-trivial bugs, also explain why the fix you provided is a good one. Sorting the bugs by what operation they were in is recommended.
3. The Debugging Process: Include things you tried that did not work and tell us about different approaches you took. If you could not find a bug that you knew was there, show us your efforts to find it. You will be able to earn partial credit from these descriptions.
4. Screenshots: Include a select few that supplement your text.
5. Solution: Please include the partially or completely decrypted text of the goal file.

Submission Details

You must submit this assignment in two ways:

1. Submit a zip file containing all the debugged source files, your finished writeup of the bugs (in .pdf format *only*), and your `README.txt` file. If you did discuss this assignment, problem solving techniques, or error messages, etc. with any course staff, please list their names in your `README.txt` file.
2. Submit a printed copy of your writeup to your TA during lab on May 11, 2016.