# CSCI-1200 Data Structures — Spring 2019
# Homework 4 Debugging (& List Iterators)

**IMPORTANT: Read this whole assignment before you download or start working on the code. The format is different from our other assignments, so be sure you have read and understood *all* of the instructions.**

Studying and documenting a software error is an essential skill that allows you to reproduce the error, fully describe the problem to other software developers on your team, and ultimately verify that your proposed code fix actually solves the problem.

In this assignment we provide source code for an existing program that decrypts an input text file. While the program is "complete" and should compile on your system, it is in need of some serious debugging. Your job is to debug this program, and then submit both your debugged version of the program *AND* a detailed writeup of how you found and fixed the bugs.

## 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 for this assignment.**

## Filing a Detailed Bug Report

To document your debugging process you should get into the habit of taking detailed step-by-step notes (and screenshots) of the errors you encounter. Consider assigning an issue or bug # to each bug you encounter; this is not required but is good practice and is how bugs are formally filed with a software project. A well-written and complete *issue or bug report* should contain the following information (as appropriate and where relevant):

- Overall development environment: What operating system? Compiler? Computer hardware? etc. (You only need to specify this once for the entire report.)
- Describe how to deliberately reproduce this bug. What were the command line arguments, input file(s), or runtime actions (keyboard input, etc.)? Does this sequence of actions always reproduce the bug?
- Describe in words the erroneous behavior: E.g., compilation error/warning, linker error, runtime crash, runtime buggy output, program hangs, etc.
- Record the exact text of any error messages.
- Record the exact output. How is this different from what you expected?
- If you are using a debugger: What file, function, and line of code? What variable had the incorrect value? etc.
- Include screenshots as appropriate to supplement the information above.

For this assignment, you should make your own notes about what tools helped you pinpoint the error. If you learned how to use a feature of your debugger, make notes for yourself on how to use that feature and why it was helpful (or not so helpful) to track down this bug.

Once you've documented the bug, fix it, and document in your report how you fixed the bug and why the fix works.

## Goal: Lots of Practice with a Step-by-Step Debugger

Tips for this assignment:

- **This is a huge, complex project.** Trying to read all of the code initially won't get you anywhere. When you pinpoint a bug, study the surrounding code as needed to understand the situation and propose a solution. If you try to make fixes blindly without understanding why they should work, you will likely cause more problems later in the program.
- **Try to avoid falling back on "Print Debugging".** Don't add `std::cout` or `printf` statements to the code. Don't comment out blocks of code. Instead, use `gdb` or your debugger of choice to navigate through the code and examine specific values in the code. Find a good reference or tutorial on your chosen debugger and learn how to use its amazing features. You may find Dr. Memory or Valgrind to be helpful as well.
- **Make your bug fixes as minimal as possible.** When working with code written by another developer, we are often tempted to make major changes because we personally would have written things differently. For this assignment, you should adopt the "If it ain't broke, don't fix it" policy. If you can see more than one valid way to fix a bug, choose the solution with the smallest edit (fewest number of lines or characters added/edited/deleted). Submitty will deduct points if your code differs from the original file too much.
- **If it says it contains no bugs, it contains no bugs.** Some sections of code or even entire functions are marked as containing no bugs. Don't waste your time on them.

## The Secret Message Decryption Program

The provided code is organized into five separate *operations* with additional helper functions. There are many intentionally-placed bugs within this code. Each operations function is designed to return a specific value that contributes to the decryption process; however, the bugs in these functions will cause the function to return the wrong value or crash! By finding and fixing all the bugs, you will decrypt the file piece by piece. The comments left behind by the original developers will tell you how the program is expected to work.

When all bugs in one operation are successfully fixed and the program is run and passed in the encrypted file as an argument, it will decrypt and print part of the contents of the file. Thus, fixing all operations completely will decrypt and print the entire file. It will be obvious when the message has been decrypted correctly — they are all plain text.

We have prepared multiple differently-buggy versions of the secret message decryption program. Login to Submitty to obtain your assigned version of the buggy decryption source code.

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

```
g++ main.cpp -lm -o decrypt.exe
```

To run the program, provide 3 arguments: a flag to indicate which operations to run, the encrypted input file, and the name of an output file where the decrypted secret message output should be written.

```
./decrypt.exe  --arithmetic-operations  encrypted_message.txt  secret_message_output.txt
./decrypt.exe  --file-operations        encrypted_message.txt  secret_message_output.txt
./decrypt.exe  --array-operations       encrypted_message.txt  secret_message_output.txt
./decrypt.exe  --vector-operations      encrypted_message.txt  secret_message_output.txt
./decrypt.exe  --list-operations        encrypted_message.txt  secret_message_output.txt
./decrypt.exe  --all-operations         encrypted_message.txt  secret_message_output.txt
```

### Your Writeup

From your detailed notes, create an approximately 1000-2000 word written report of the debugging process. Rather than detail every bug found and fixed, choose 5-10 bugs that are interesting and demonstrate a variety of debugging methods. Try to show off each of the debugger skills listed above in at least one of these bug plus bug fix descriptions. *NOTE: We're more interested in your thought process, the description of your debugging process, and the new skills you learned, than we are in the formal bug report and code patch.*

You may include screenshots or paste code blocks. But your report should be concise and well-written. You should not have font smaller than 10 point font, and your writeup should be no more than 6 pages in length. You should format this report as a `.pdf` document. We will not accept or grade any other file format for the report. **The graders will stop reading after 6 pages, and will not use word count for grading.**

### Grading Criteria

Your grade for this assignment will be split into the following components:

**15 pts** Successful decryption of the secret message. (Partial credit will be awarded for partial decryption.)

**10 pts** Source code changes to fix bugs are small. (See the "Rules".)

**25 pts** Writing Quality:

- Report organization and structure: Don't just submit a full chronological transcript of your debugging process. If you find a bug that is very similar to another bug, spend no more than a sentence on noting it.
- Readability: Be concise and use complete sentences. Avoid spelling and grammar mistakes.
- Clarity: Describe each selected bug and why it is causing the program to fail. Describe how you found the problem. For non-trivial bugs, also explain why the fix you provided is a good one.
- 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.
- Screenshots of Debugger Use: Include a select few that supplement your text. Don't include multiple screenshots of every single bug you found, and don't use screenshots to eat up lots of space (longer reports do not guarantee higher grades, graders will stop reading after 6 pages).

### Wednesday Night Extension

If you earn at least 6 points on Test Cases 4-8 by Wednesday February 13th 11:59:59PM, you can submit on Friday without using a late day. Even with the extension and late days, no submission will be accepted after Saturday February 16th 11:59:59PM.