Gabriel Maayan

Professors Holzbauer and Thompson

Data Structures

2/23/17

Homework 4 Bug Report

All of the errors in this report were found on a computer running Windows 10 with a x64 based processor and 7.42 GB usable RAM. The program was compiled with g++ version 5.4.0-1 using the Cygwin64 command line.

The first few errors in the homework were all simple arithmetic errors. While most of these are easy to spot just looking through the code, it is often easier to use the debugger to watch certain variables to know what needs changing. To do this, I compiled the program with g++, adding –g, and then started the gdb debugger. I set up a breakpoint at arithmetic_operations() with "break arithmetic_operations," then I ran the program with "run –arithmetic-operations encrypted_file.txt out.txt." Once I reached the breakpoint, I used "display e" to watch the "e" variable. Stepping through the function with "next," I noticed "e" was set to 36 instead of 32 like the comments asked for on line 287. The simple fix for this was to change "int e=b-3*a+5*c;" to "int e=b-3*a+4*c;" on line 287.

I found some other errors just by compiling the program using g++ with all of the warnings enabled:

```
$ g++ -Wall -g operations.cpp -lm -o decrypt.exe
operations.cpp: In function 'int vector_operations()':
operations.cpp:252:36: warning: for increment expression has no effect [-Wunused-value]
   for(uint i = 0; i < all.size(); i+1) {
                                    ^
operations.cpp: In function 'int list_operations()':
operations.cpp:506:19: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
     for(int i=0; i<itr->size(); ++i) {
                  ^
operations.cpp: In function 'int pythagoras(int, int)':
operations.cpp:63:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
operations.cpp:52:24: warning: 'placeholder' is used uninitialized in this function [-Wuninitialized]
   float fracpart = modf(sqrt(sumsquares), placeholder);
                        ^
operations.cpp: In function 'bool file_operations(int, char**, char*&, int&)':
operations.cpp:369:33: warning: 'length' may be used uninitialized in this function [-Wmaybe-uninitialized]
   char* buffer = new char[length];
                                 ^
```

An interesting error that I found using this was that the function “int pythagoras(int, int)” was “[reaching] end of non-void function.” Looking at the code for this function, I realized that all of the return statements were in conditionals so the program might reach the end of the function without finding anything to return. The fix for this was to add “return -1;” on line 62. The -1 is because that is what the comments ask to be returned if it reaches the end of the function.

A very good example of an error that can be found just by inspection was in the “file_operations()” function, there was some faulty error checking. It was easy to see on line 360 that “if(infile)” doesn’t do anything. To fix this, following the guide on the course website, I changed it to “if(!infile.good()),” which solved the problem.

Another way I find very useful to debug code is a memory debugger, specifically Dr. Memory in my case. I find most errors very easy to read and it is helpful to see all of the line numbers where the error might have originated.

```
~~Dr.M~~ Error #1: UNINITIALIZED READ: reading register eax
~~Dr.M~~ # 0 modf                        [/usr/src/debug/mingw64-i686-runtime-5.0.1-1/math/modf.c:38]
~~Dr.M~~ # 1 pythagoras                  [/home/maayag/ds/hw4/operations.cpp:52]
~~Dr.M~~ # 2 array_operations            [/home/maayag/ds/hw4/operations.cpp:84]
~~Dr.M~~ # 3 main                        [/home/maayag/ds/hw4/operations.cpp:616]
~~Dr.M~~
~~Dr.M~~ Error #2: WARNING: writing to readonly memory 0x738652a5-0x738652ad
~~Dr.M~~ # 0 modf                        [/usr/src/debug/mingw64-i686-runtime-5.0.1-1/math/modf.c:39]
~~Dr.M~~ # 1 pythagoras                  [/home/maayag/ds/hw4/operations.cpp:52]
~~Dr.M~~ # 2 array_operations            [/home/maayag/ds/hw4/operations.cpp:84]
~~Dr.M~~ # 3 main                        [/home/maayag/ds/hw4/operations.cpp:616]
```

The first error here was because “placeholder” was never allocated any memory in “pythagoras().” The fix for this is to change line 47 from “double* placeholder;” to “double* placeholder = new double();” This creates a memory leak, also shown by Dr. Memory, because “placeholder” is never deleted. The way I fixed this was to make an int to hold “*placeholder,” then “delete placeholder;,” and then return the int. This code was placed on lines 54 and 63. Additionally, I simply put “delete placeholder;” on line 67 before the final return statement.

On line 60 there was a math error that I only found after using gdb to examine variables like I did above. Using the same procedure, I found that “float diffsquares = y*y - x*x;” only worked when y was the hypotenuse, because it would give a negative value if x was the hypotenuse. I fixed this problem by changing the line to “float diffsquares = abs(y*y - x*x);” Now that the function took the absolute value, it would give the correct answer no matter which variable was the hypotenuse. As above, in gdb, I used “break array_operations()” and then “display array[5][4]” to find the problem.

Through just normal running the program with Cygwin, I found a lot of errors that were errors simply because the program wasn’t doing what the comments said it should. These I found just by assertions that failed or by reading through snippets of code. On line 408, “int vector_sum(std::vector<int> inVec)” the comments said that “inVec” should be modified by the function “vector_sum,” but here it is passed in by value. To fix this, I changed line 408 to “int vector_sum(std::vector<int>& inVec).”

Another error I found using Dr. Memory was on line 457:

```
$ drmemory -brief -batch -- decrypt.exe --list-operations encrypted_message.txt out.txt
~~Dr.M~~ Dr. Memory version 1.11.0
~~Dr.M~~ Running "decrypt.exe --list-operations encrypted_message.txt out.txt"
~~Dr.M~~
~~Dr.M~~ Error #1: UNADDRESSABLE ACCESS of freed memory: reading 4 byte(s)
~~Dr.M~~ # 0 std::_List_iterator<>::operator--              [/usr/lib/gcc/i686-w64-mingw32/5.4.0/include
/c++/bits/stl_list.h:182]
~~Dr.M~~ # 1 list_operations                                [/home/maayag/ds/hw4/operations.cpp:457]
~~Dr.M~~ # 2 main                                           [/home/maayag/ds/hw4/operations.cpp:642]
```

I knew it had something to do with the iterator getting deleted, but I wasn't sure why it was happening. I went to "http://www.cplusplus.com/reference/list/list/erase/" to read about the list "erase()" function that was being called, and I found out that iterators that are pointing to the erased element also get deleted, so "l500.erase(itr);" made "itr" unusable after the call. To fix this I changed line 457 to "itr = l500.erase(itr);" and everything worked.

Strangely, I found that using gdb didn't work in some of the later operations, specifically "list_operations()" and I am still not sure why. I compiled the program with g++ and –g, and started gdb. I set up a breakpoint at list_operations with "break list_operations()," and ran the program with "run –list-operations encrypted_message.txt out.txt." The program starts and exits without ever reaching the breakpoint. I also tried setting up an additional breakpoint at main and then stepping through the program, but as soon as it hit "list_operations()," it would exit without stopping. While I was able to use normal debugging and Dr. Memory to find and fix all of the remaining errors, I'm still not sure why this happened and would have liked to use gdb in "list_operations()." I found that Dr. Memory worked very well for me in nearly all cases. Even though it would not allow me to view individual variables or step through the program, it did show me all of the output like normal running the program, and it showed me all of the errors along with a description of what went wrong and the line numbers of where the error was created going back to main(). As I mentioned before though, it is only good at finding actual errors in the code, it's not helpful in finding differences in how the program is supposed to run with how it actually runs.

```
$ gdb decrypt.exe
GNU gdb (GDB) (Cygwin 7.10.1-1) 7.10.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-pc-cygwin".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from decrypt.exe...done.
(gdb) break list_operations()
Breakpoint 1 at 0x10040313a: file operations.cpp, line 433.
(gdb) run decrypt.exe --list-operations encrypted_message.txt out.txt
Starting program: /cygdrive/c/Users/maayag/Documents/DataStructures/hw4/decrypt.exe decrypt.exe --list-op
erations encrypted_message.txt out.txt
[New Thread 4456.0x3288]
[New Thread 4456.0x1ef4]
[New Thread 4456.0x25a0]
[New Thread 4456.0x30f4]
[New Thread 4456.0x2484]
[Thread 4456.0x25a0 exited with code 0]
[Thread 4456.0x30f4 exited with code 0]
[Thread 4456.0x1ef4 exited with code 0]
[Inferior 1 (process 4456) exited normally]
```