Development Environment

Computer and Hardware

**MacOS High Sierra** Version 10.13.3

MacBook Pro (15-inch, 2016)

Processor 2.7 GHz Intel Core i7

Memory 16GB 2133 MHz LPDDR3

Radeon Pro 455 2 GB

Compiler (LLVM version 8.0.0 (clang-800.0.42.1))

Configured with: --prefix=/Library/Developer/CommandLineTools/usr --with-gxx-include-dir=/usr/include/c++/4.2.1

Apple LLVM version 8.0.0 (clang-800.0.42.1)

Target: x86\_64-apple-darwin17.4.0

Thread model: posix

IDE

Xcode 9.2

Build version 9C40b

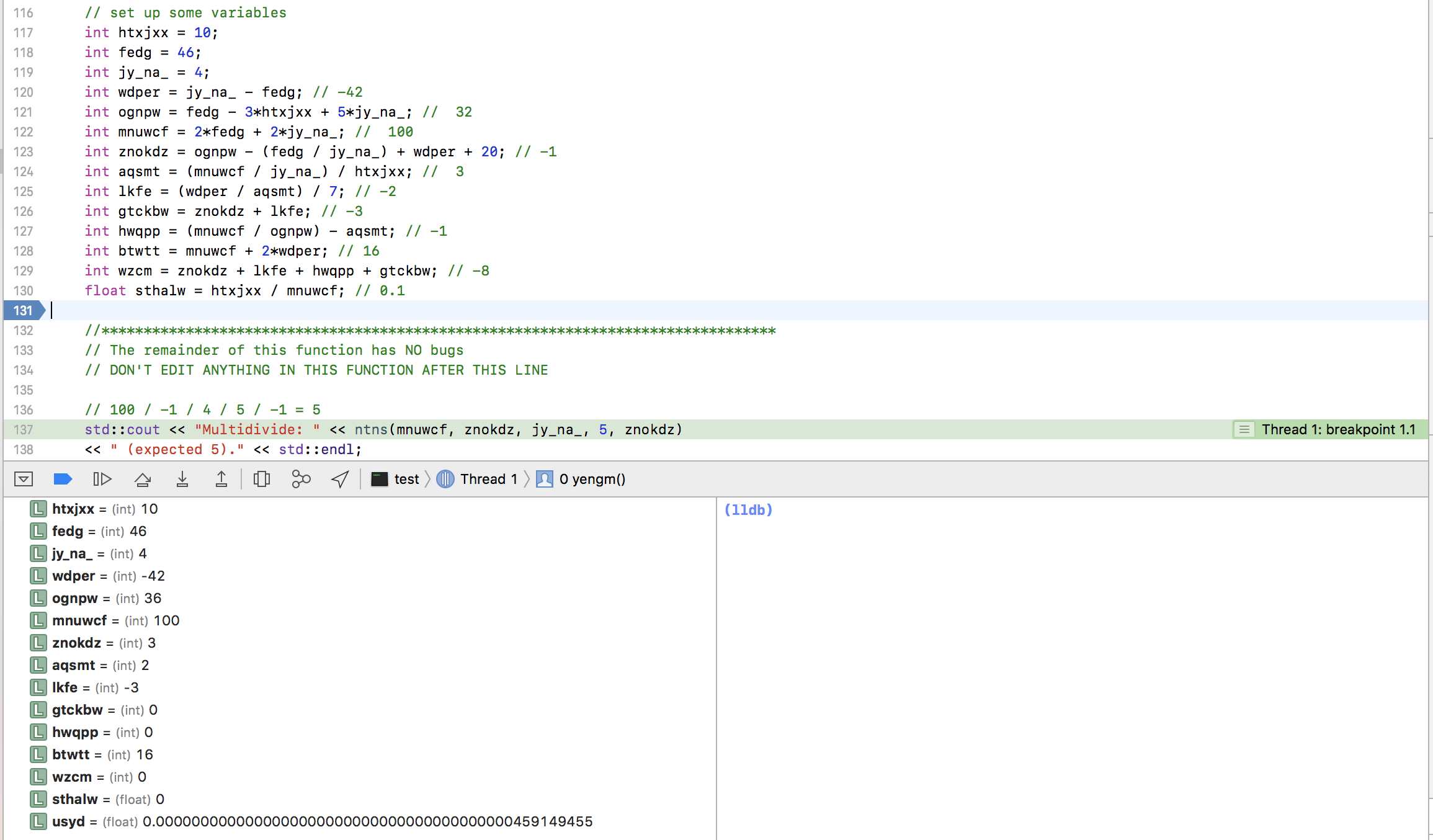
The program is a c++ document that will read in arguments to execute specific functions, read in an encrypted file and output the decrypted text. The total program mainly contains 6 larger parts of which 5 of the different operations contained various forms of, mainly, semantic errors. Each of these operations can be tested quite independently and they show a rough similarity in the types of errors they experience in each function which I hypothesized may be due to the data type being handled as they handle different types in each operation. This potentially gives a good indicator of the types of errors that appear while using the various types of data so I chose to explore the program through the operations rather than the individual bugs. All bugs can be reproduced by running the compiling the file and running the compiled executable with the arguments:

./decrypt.exe --all-operations encrypted\_message.txt secret\_message\_output.txt

Similarly, bugs in specific parts of the program can be found and reproduced by running the executable with the specific argument for that operation.

**Bug ID#: Arithmetic-01**

* Summary:
* Details/Steps: 
  + g++ main.cpp –o decrypt.exe -Wall
  + ./decrypt.exe --arithmetic-operations encrypted\_message.txt secret\_message\_output.txt
* Error Type: Semantic Error
* Errors Desc or message or output:
  + **Assertion failed: (ntns(mnuwcf,znokdz,jy\_na\_,5,znokdz) == 5), function yengm, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 132.**



* Tools used: lldb debugger
* Fix Decription : The variables, ognpw, aqsmt, hwqpp, wzcm and sthalw, being calculated did not match their desired values according to the debugger and I adjusted the functions to find their values accordingly. Among these, the first four were due to arithmetic errors while the last was a computational error due to not having a float among the variables being divided.
* Comments: The first four variables were fixed through the manipulation of constants while the last was fixed by converting one of the two variables which decided its value to a float inside the assignment.

Bugs in arithmetic operation:

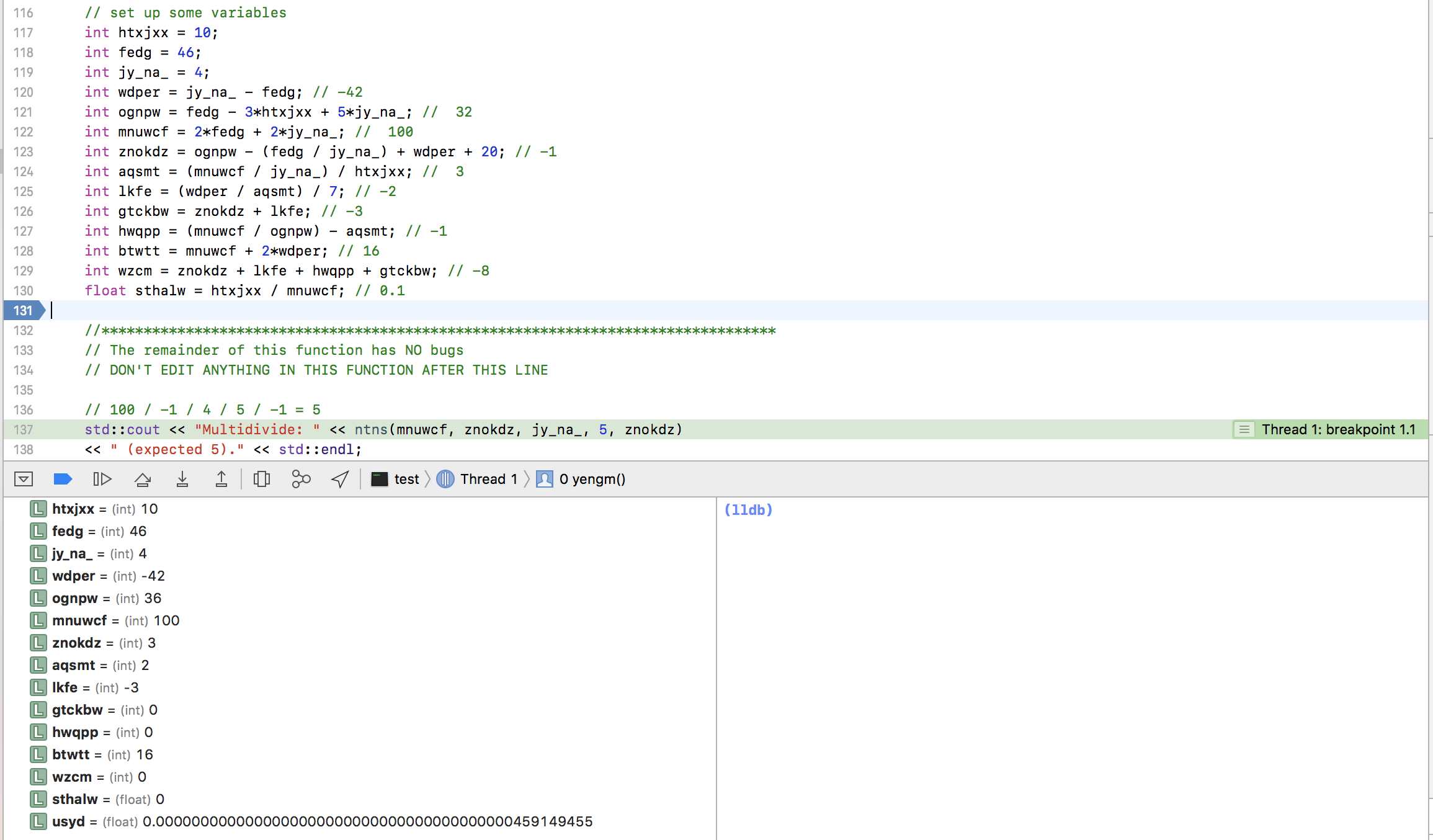
./decrypt.exe --arithmetic-operations encrypted\_message.txt secret\_message\_output.txt

The errors in this part of the program are entirely due to the presence of semantic errors and problems in computation between different memory types. I solved these errors in two steps due to the presence of two distinctly different types of errors.

Bug Type 1-

**Assertion failed: (ntns(mnuwcf,znokdz,jy\_na\_,5,znokdz) == 5), function yengm, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 132.**

The first was the fact that the variables being calculated did not match their desired values using the debugger and adjusted the functions to find their values accordingly. The variables ognpw, aqsmt, hwqpp, wzcm and sthalw. Among these, the first four were due to arithmetic errors while the last was a computational error due to not having a float among the variables being divided.



Bug Type 2-

**Multidivide: 0 (expected 0.1).**

**Assertion failed: (cvljhh(usyd, sthalw)), function yengm, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 157.**

The second type of error in the arithmetic operation was when executing the ntns function which consisted of arithmetic between a number of int types being converted to a float. This type of computation would yield an int in place of a float and was found through the use of a breakpoint inside the ntns function on line 161.

Bugs in file operations:

./decrypt.exe --file-operations encrypted\_message.txt secret\_message\_output.txt

**Usage: /Users/sriyuthsagi/Library/Developer/Xcode/DerivedData/test-aunpkkrglxassvgzztnqbnxctvgr/Build/Products/Debug/test operations infile outfile**

**Couldn't start operations. -Line 43**

**That file could not be opened! -Line 55**

File operations held two errors, however, both these errors were purely semantic and were due to the incorrect return of Boolean statements due to not using inequality operators properly. The bugs were found on lines 43 and 55 and were logical errors that could be avoided with an inequality operator (!).

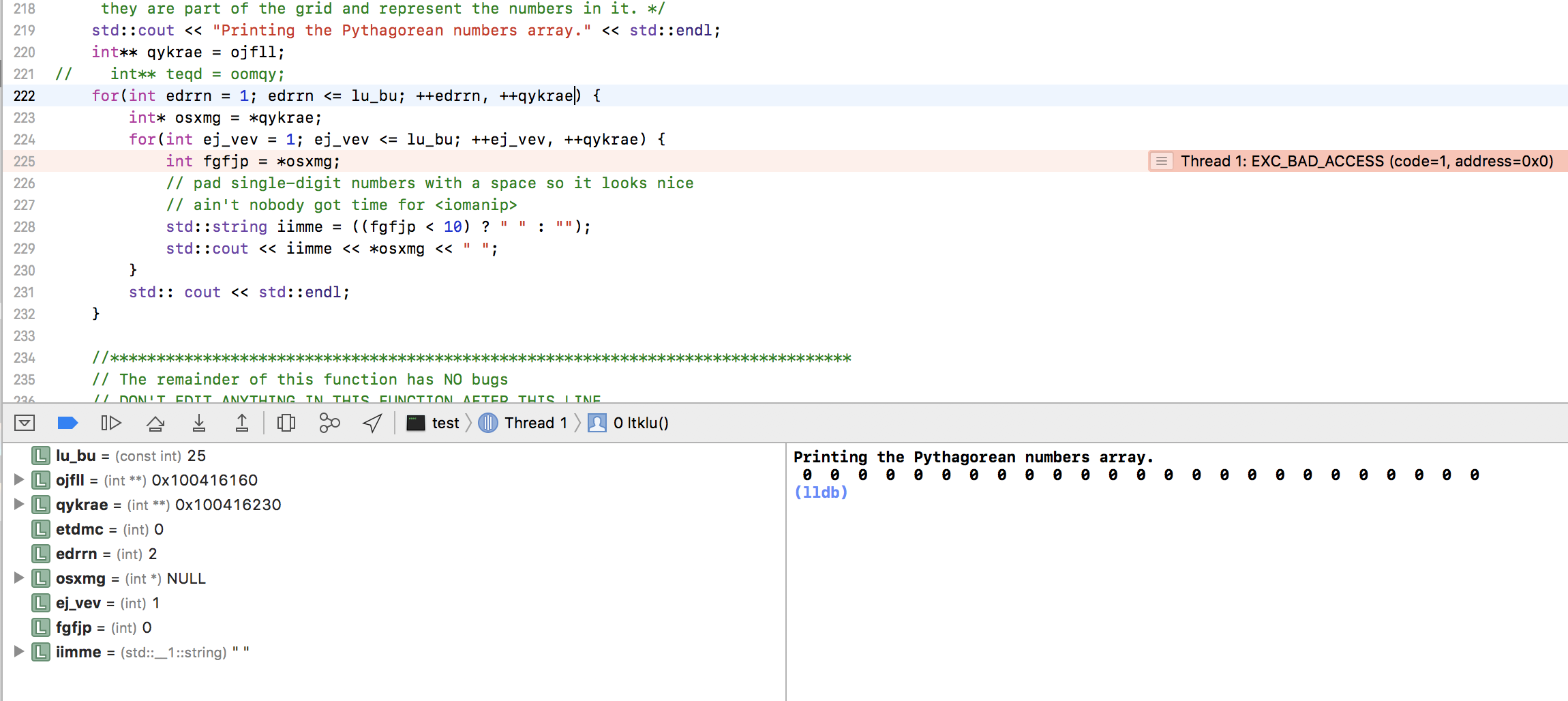
Bugs in array operations:

./decrypt.exe --array-operations encrypted\_message.txt secret\_message\_output.txt

Array operations held mainly two different types of bugs. One of these is the presence of errors with defining boundaries which results in problems in calling values from the array while the other bug lies in the use of an array in the function blq\_\_ without deleting the array afterwards or using its intended functionality as an array

Bug Type 1-

Thread 1: EXC\_BAD\_ACCESS (code=1, address=0x0)



These types of bugs are not due to any misunderstandings in the code itself and show the importance of understanding where values are being called from. After locating the error, EXC\_BAD\_ACCESS (code=1, address=0x0) shows that the variable is trying to call on a value that was not defined and this type of bug can be fixed by redefining the boundaries.

Bug Type 2-

**Assertion failed: (ojfll[1][2] == -1), function ltklu, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 203.**

This bug was a result of a variety of problems present in the blq\_\_ function that was called previously. One of it’s problems was the most straightforward problem in the homework as it was the fact that it was possible to fail all the if statements and not return anything. This was fixed by adding a return -1; to the end of the function. The second problem was the use of an array \*hqahy to compare values and check if the other values are part of a Pythagorean triple. The problem here came about with problems in the use of pointers and the fact that an array was not necessary in this instance. To fix this, I changed the array \*hqahy to a normal float hqahy which better fit in with the rest of the problem and conversely changed the reference and value calls from the float. Furthermore there were problems in the use of the equality operator in both if statements as an assignment operator was presented instead of an equality operator. The last major logical error in the function was on line 407 where the function tested if x was the hypotenuse but failed to test if y was a possible hypotenuse which was resolved through the use of an abs statement which would be able to check if either are true.

Bugs in vector operations:

./decrypt.exe --vector-operations encrypted\_message.txt secret\_message\_output.txt

Bug Type 1-

Thread 1: EXC\_BAD\_ACCESS (code=1, address=0x50221196c)