Bug #1

The error message I got when I ran arithmetic operations was, Assertion failed: (multidivide(f,g,c,5,g) == 5), function arithmetic_operations, file operations_unfixed.cpp, line 128. The way I went about trying to fix this bug was using lldb. I set a breakpoint at the function arithmetic_operations and stepped through it. When I got to the place where the variable s is declared, I used the frame variable command to print out all the values in the scope.

```
(lldb) frame variable
(int) a = 10
(int) b = 46
(int) c = 4
(int) d = -42
(int) e = 36
(int) f = 100
(int) g = 3
(int) h = 2
(int) m = -3
(int) n = 0
(int) p = 0
(int) q = 16
(int) r = 0
(float) s = 2.76643107E+19
(float) zeropointone = 0
```

```
int arithmetic_operations() {

    // set up some variables
    int a = 10;
    int b = 46;
    int c = 4;
    int d = c - b;                  // -42
    int e = b - 3*a + 5*c;          //  32
    int f = 2*b + 2*c;              //  100
    int g = e - (b/c) + d + 20; // -1
    int h = (f/c) / a;              //  3
    int m = (d / h) / 7;            // -2
    int n = g + m;                  // -3
    int p = (f / e) - h;            // -1
    int q = f + 2*d;                // 16
    int r = g + m + p + n;          // -8
    float s = a / f;                //  0.1
```

```
// set up some variables
int a = 10;
int b = 46;
int c = 4;
int d = c - b;                  // -42
int e = b - 3*a + 4*c;          //  32
int f = 2*b + 2*c;              //  100
int g = e - (b/c) + d + 20; // -1
int h = ceil(double((f/c)) / a); //  3
int m = (d / h) / 7;            // -2
int n = g + m;                  // -3
int p = (f / e)-1 - h;          // -1
int q = f + 2*d;                // 16
int r = g + m + p + n-1;        // -8
float s = float(a) / f;         //  0.1
```

I realized that the values being calculated weren't equal to the expected values. So I went through each variable that was not equal to the expected value and made the necessary changes.

Bug #2-

An error I encounter when running file_operations is a memory leak of 131 bytes which pointed me to the varialbe buffer. Although buffer was not able to be delete because it is used after the place in the code where I was able to edit the code and it goes out of scope after the function returns. But buffer is set equal to returned_buffer, which is passed by reference and so I was able to delete in main where return_buffer is also in scope.

```
// (Hopefully) decrypt and print out the file!
// It should work if you get all the FIXED messages!
if(ops == "--all-operations") { // only do it if testing everything
  if(decrypt(records, file_buffer, file_buffer_length, outFileName)) {
    std::cout << std::endl << "Decryption successful - good job!" << std::endl <<
        std::endl;
     delete [] file_buffer;
    return 0;
  } else {
    std::cout << "Decryption failed or was incomplete" << std::endl;
    return 1;
```

```
Bug #2 - Memory
~~Dr.M~~ Error #1: LEAK 131 bytes
~~Dr.M~~ # 0 replace_operator_new_array    [/drmemory_package/common/alloc_replace.c:292
~~Dr.M~~ # 1 file_operations               [/Users/Mike/Dropbox/Data
Structures/hws/hw4code/hw4code/operations.cpp:75]
~~Dr.M~~ # 2 main                          [/Users/Mike/Dropbox/Data
Structures/hws/hw4code/hw4code/operations.cpp:608]
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~     0 unique,    0 total unaddressable access(es)
~~Dr.M~~     0 unique,    0 total uninitialized access(es)
~~Dr.M~~     0 unique,    0 total invalid heap argument(s)
~~Dr.M~~     0 unique,    0 total warning(s)
~~Dr.M~~     1 unique,    1 total,   131 byte(s) of leak(s)
~~Dr.M~~     0 unique,    0 total,     0 byte(s) of possible leak(s)
```

I also had to delete it in the if statement in main that check if the terminal argument is –file-operations, but when it is no –all-operations.

Bug #3-

When running –array-operations, I set a break point at array_opertations, because the numbers in the array of pytharom squares were not correct, so I step through and realize the it was occuring in the function pythagorus. Pythagorus had a few errors. If we look up the correct way we use modf, we will see that the parameters for the function are two doubles, the second one passed by reference, not a pointer to a double. Another error in this function was if x was the hypotonus, then diffsquares will be negative and the square root of a negative number are imagenary, so you need to make it the absolute value of the y^2 – x^x. The last error/ warning

was there was no default return for if no third side was found, so I had to add the return -1 statement.

```
Process 37345 stopped
* thread #1: tid = 0x244e4c, 0x000029e2 test.exe`pythagoras(x=0, y=0) + 50 at
operations.cpp:187, queue = 'com.apple.main-thread', stop reason = step over
    frame #0: 0x000029e2 test.exe`pythagoras(x=0, y=0) + 50 at operations.cpp:187
   184
   185     // x and y are both legs
   186     float sumsquares = (x*x) + (y*y);
-> 187     float fracpart = modf(sqrt(sumsquares), placeholder);
   188     if(fracpart == 0)
   189       return (int) *placeholder;
   190
(lldb) s
Process 37345 stopped
* thread #1: tid = 0x244e4c, 0x9041a77b libsystem_m.dylib`modf + 107, queue =
'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x0)
    frame #0: 0x9041a77b libsystem_m.dylib`modf + 107
libsystem_m.dylib`modf + 107:
-> 0x9041a77b:  movsd  %xmm3, (%ecx)
   0x9041a77f:  fldl   0x4(%esp)
   0x9041a783:  retl
   0x9041a784:  ucomisd %xmm0, %xmm0
```

Bug #4-

When running the argument --array-operations, I received a segmentation fault 10. So using lldb, I stepped through to find it happening somewhere on lines 365-368. First thing I change was the bounds of the for loo was running form x,y = 1 to x,y >= size. So the the array was accessing memory that was not allocated. So then I ran drmemory and receive an UNADDRESSABLE ASSESS error. I fixed it by stopping increment the pointer and incrementing only x and y while using pointer indexing.

```
std::cout << "Printing the Pythagorean numbers array." << std::endl;
int** tmp_ptr = array;
for(int x = 1; x <= size; ++x, ++tmp_ptr) {
    int* tmp_ptr2 = *tmp_ptr;
    for(int y = 1; y <= size; ++y, ++tmp_ptr) {
        int tmp = *tmp_ptr2;
        // pad single-digit numbers with a space so it looks nice
        // ain't nobody got time for <iomanip>
        std::string maybe_space = ((tmp < 10 && tmp >= 0) ? " " : "");
        std::cout << maybe_space << *tmp_ptr2 << " ";
    }
```

```
   they are part of the grid and represent the numbers in it. */
itd::cout << "Printing the Pythagorean numbers array." << std::endl;
int** tmp_ptr = array;
for(int x = 0; x < size; ++x) {
  int* tmp_ptr2 = *(tmp_ptr+x);
  for(int y = 0; y < size; ++y) {
    int tmp = *(tmp_ptr2+y);
    // pad single-d   igit numbers with a space so it looks nice
    // ain't nobody got time for <iomanip>
    std::string maybe_space = ((tmp < 10 && tmp >= 0) ? " " : "");
    std::cout << maybe_space << tmp << " ";
  }
```

Bug #5-

When running –vector-operations, I got an arror saying, Assertion failed: (v1sum == 175), function vector_operations, file operations.cpp, line 420. Abort trap: 6, and also a drmemory error

```
~~Dr.M~~ Error #1: UNADDRESSABLE ACCESS beyond heap bounds: reading 4 byte(s)
~~Dr.M~~ # 0 vector_sum                    [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:520]
~~Dr.M~~ # 1 vector_operations             [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:412]
~~Dr.M~~ # 2 main                          [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:624]
~~Dr.M~~
~~Dr.M~~ Error #2: UNADDRESSABLE ACCESS beyond heap bounds: reading 4 byte(s)
~~Dr.M~~ # 0 vector_sum                    [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:520]
~~Dr.M~~ # 1 vector_operations             [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:412]
~~Dr.M~~ # 2 main                          [/Users/Mike/Dropbox/Data Structure
s/hws/hw4code/hw4code/operations_unfixed.cpp:624]
~~Dr.M~~ Note: refers to 0 byte(s) beyond last valid byte in prior malloc
```

Which pointed to the vector sum function that was giving us these problems. By reading the comments that tells you what the exact things the function should do, you can tell that the contents of inVec should be modified, but is not returned, so you must make inVec passed in by reference so it can be changed and v1sum can be correxct. For the memory error, we can see that the for loop is going out of bouds because it is going to i<=inVec.size(), so I change the <= to <. Also in the assertion for the vector_compare falied, so I set a break point in the function. The function did not have any check for if the vectors have different sizes. So in the for loop, I added the condition of to check if i is less than the size of the v1 vector in addition to checking if i is less than the size of v2 vector size.

Bug #6-

When running the arguments for –list-operations, I set a breakpoint at list_operations, the problem was the alphebey in list l1, the for loop was pushing the upercases to the from so it created the uppercase letters backwards and I change the c value to subtract instead of add. On lines 223, the erase function wasn't earsing correctly, so inorrder to erase, I set erase called equal to the iterator variable itr, and then –itr. Also change the not equal signs in the if statements to equal statements. On lines 266-269, there were memory errors. I switch the reverse iterator in the for loop to a iterator, and changed the ++fruit_itr to just fruit_itr in the erase function call, the --ritr to decrement the iterator. Another error was that "uglyfriut" was push on to the list twice, so to fix this, I had to delete on of the puch statements with "uglyfruit" that was in the wrong spot.

Bug #7-

There were also 7 warning that needed to be fixed that were show when including –Wall. The warning was -operations.cpp:487:36: warning: expression result unused [-Wunused-value] for(uint i = 0; i < all.size(); i+1) {. The fix was changing the i+1 to i++ because the for loop would not be used if i+1 is there because it does not do anything. So it needed to be changed to i++ so the for loop could be used and incremented correctely. The next warning was operations.cpp:498:27: warning: comparison of unsigned expression >= 0 is always true [-Wtautological-compare] for(uint i=counter-1; i >= 0; --i) {. This is a problem because the bounds go below 0 and cause a potential error also uint's cannot go below zero. The fix for this is changing the greater or equal symbol to a greater than symbol. The next warning was operations.cpp:64:27: warning: variable 'length' is uninitialized when used here [-Wuninitialized] char* buffer = new char[length]; The fix to this was moving the line, char* buffer = new char[length];, after where length is given a value. The next warning is operations.cpp:437:7: warning: variable 'counter' is uninitialized when used here [-Wuninitialized] counter++;. The fix is to set counter equal to zero in order to initialize the value of counter.