# Bug report

Overall development environment:

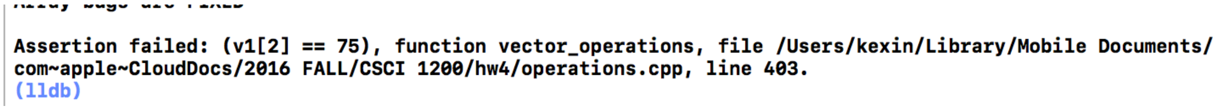
Xcode as IDE, c++98 version. During the dubug process, also use DrMemory to debug the memory error.

BUG#1 and #2 are both about wrong values in containers. #3 and #4 involves in pointer. #5 and #6 are trivial problems.

BUG #1: line 403. In the function, vector parameter should be call by reference

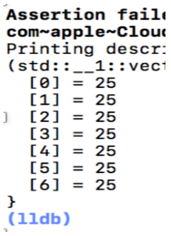
*How to find?*

Error message below told me that assert failed.



```
Assertion failed: (v1[2] == 75), function vector_operations, file /Users/kexin/Library/Mobile Documents/
com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp, line 403.
(lldb)
```

Then I use the debugger to find out what is in vector v1.



```
Assertion faile
com~apple~Clouc
Printing descri
(std::__1::vect
  [0] = 25
  [1] = 25
  [2] = 25
  [3] = 25
  [4] = 25
  [5] = 25
  [6] = 25
}
(lldb)
```

So, there must be some mistakes in the function **vector_sum(std::vector<int> inVec),** since this is the only function to modify v1.(Before that, there is a bug wrong initialized of index i, should fix it first). According to the comment above,  I find out that the vector is modified inside the function. But when it is outside the function, the value in the function does not change.

*How to fix?*

Add & just let the parameter call by reference, then it will change outside the function.

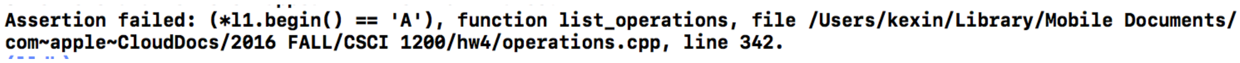
**vector_sum(std::vector<int> &inVec),** *don't forget to change the function prototypes, they must be the same.*

*Why important?*

It is important to understand the type of parameter when create a function. To make sure whether the parameter should be modified after function or not.

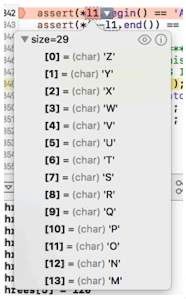
BUG #2 line 241. In function array_operation()

*How to find?*



```
Assertion failed: (*l1.begin() == 'A'), function list_operations, file /Users/kexin/Library/Mobile Documents/
com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp, line 342.
```

Error message above told me that assert failed. So Use Xcode to find out what is in l500.



```
assert(*l1.begin() == 'A
assert(*--l1.end()) ==
▼ size=29
    [0] = (char) 'Z'
    [1] = (char) 'Y'
    [2] = (char) 'X'
    [3] = (char) 'W'
    [4] = (char) 'V'
    [5] = (char) 'U'
    [6] = (char) 'T'
    [7] = (char) 'S'
    [8] = (char) 'R'
    [9] = (char) 'Q'
    [10] = (char) 'P'
    [11] = (char) 'O'
    [12] = (char) 'N'
    [13] = (char) 'M'
```

It looks like the capital letters are in the reversed order. Back to line the for loop where create the l500 list for the Capital letters (line 241-243). Notice that the code push_front, there is a logic error here, when push_front the letters, they will be in reversed order.

*How to fix?*

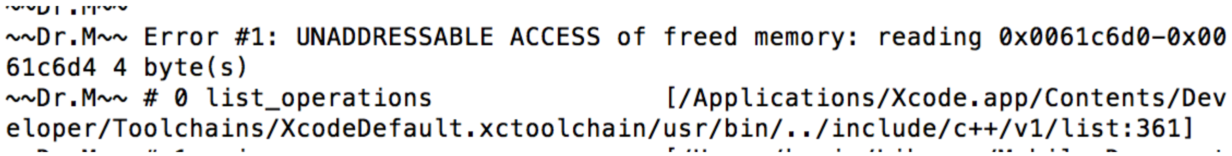
Change the for loop statement from **for(char c = 'A'; c >= 'Z'; c++)** to **for(char c = 'Z'; c >= 'A'; c--).** Push_front the letters in reversed order, then the output will be correct.

*Why important?*

When use a code that you are not very familiar, it is important to find out how it should be use. Also, when write the statement in a loop, it is crucial to understand how the for loop will continue.

BUG #3: line 262 in function list_operation()

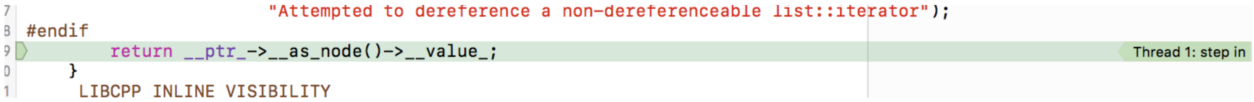
*How to find?* It looks like good in my Xcode, no error message and can run successfully. However, when it run in DrMemory, there will be error message.



```
~~Dr.M~~ Error #1: UNADDRESSABLE ACCESS of freed memory: reading 0x0061c6d0-0x00
61c6d4 4 byte(s)
~~Dr.M~~ # 0 list_operations               [/Applications/Xcode.app/Contents/Dev
eloper/Toolchains/XcodeDefault.xctoolchain/usr/bin/../include/c++/v1/list:361]
```

DrMemory told me that error occurs in 361. The for loop looks fine to me. Then set two breakpoints before and after the for loop, looks nothing wrong.

Then I started to consider the vector involved in the for loop **l500.** Back to line 262 and set some breakpoints. Step in at a time. Then it will step into another cpp file and have the message below.



```
                        "Attempted to dereference a non-dereferenceable list::iterator");
#endif
        return __ptr_->__as_node()->__value_;
    }
    LIBCPP INLINE VISIBILITY
```

Thread 1: step in

There is something wrong with the iterator.

*How to fix?*

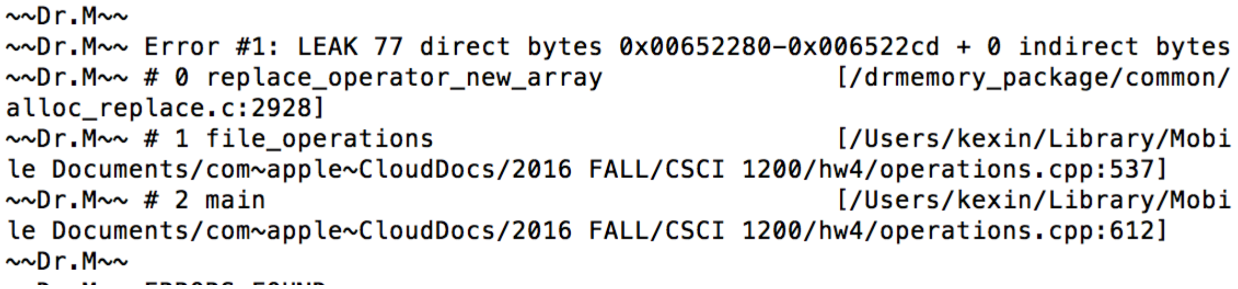
When erase something in the list, the iterator will be immediately invalid so change it from **l500.erase(itr)** to **itr = l500.erase(itr)**; every time, itr will be assign a new value which is exactly the former value, then it will not be invalid.

*Why important?*

It is important to understand the property of iterator and understand how to set breakpoint and step in. To find out the problems that cannot tell in the surface.

BUG #4: Line 654. In the main function

*How to find?*



```
~~Dr.M~~
~~Dr.M~~ Error #1: LEAK 77 direct bytes 0x00652280-0x006522cd + 0 indirect bytes
~~Dr.M~~ # 0 replace_operator_new_array               [/drmemory_package/common/
alloc_replace.c:2928]
~~Dr.M~~ # 1 file_operations                          [/Users/kexin/Library/Mobi
le Documents/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:537]
~~Dr.M~~ # 2 main                                     [/Users/kexin/Library/Mobi
le Documents/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:612]
~~Dr.M~~
```

DrMemory print the error message that there is memory leak.

I find out that in line 537, new array is built on the heap, so it may be a problem. Or in line 612 it calls file_operations(argc, argv, file_buffer, file_buffer_length), the parameter file_buffer is a pointer and build something on the heap. Also, according to the comment, file_buffer holds array of raw bytes, which should be 77 bytes when we read in the file.

*How to fix?*

It is difficult to find out the leak. Luckily, there is only two places that will happen memory leak.

In line around 537, I found out there is no place to delete buffer. It will be used until the function end. I tried to delete buffer array before return function, then some message in decrypted file will be missing. So it cannot be deleted. Then I find out that it will be used later so if it is deleted, values will be missing. So I delete file_buffer it before the whole function return to make sure it is no more than needed, then there is no error message in DrMemory.

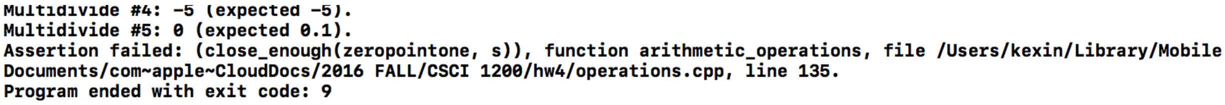
**delete [] file_buffer** in line 654

*Why important?*

I tried for a long time to find the memory leak, it is difficult to find out where is the exact location memory leak. I am not good at it, so I think it is important for me in future work to understand it more.

BUG #5: line 131. In the function arithmetic operations

*How to find?*



```
Multidivide #4: -5 (expected -5).
Multidivide #5: 0 (expected 0.1).
Assertion failed: (close_enough(zeropointone, s)), function arithmetic_operations, file /Users/kexin/Library/Mobile
Documents/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp, line 135.
Program ended with exit code: 9
```

The error message told me that assertion failed, so the close_enough(zeropointone, s) is wrong. From the output above, I can find out that the multidivide #5 is different from the expected value. Then back to line 132, where cout the value. So I can find that zeropointone should be 0.1 not 0. From the comment I can see that zeropointone should be the value 1000/10/10/10/10. It calls the multidivide function be get the value 0. Back to the multidivide (int numerator, int d1, int d2, int d3, int d4) function I can see that all the type of the divisors are int, which means it will not get a float for the answer, because int/int only get the biggest int part for the result.

*How to fix?*

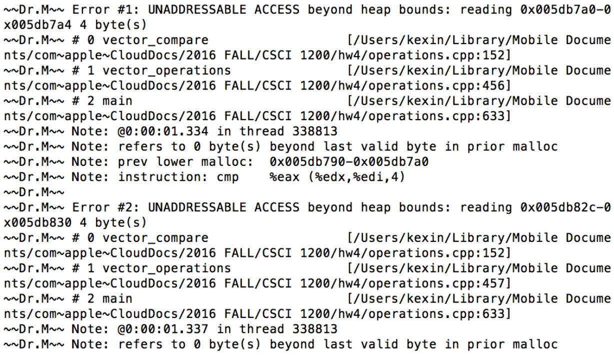
Because the bug happened at the d4 1000/10/10/10=1. 1/10 should be 0.1 if it is a pure mathematics calculation. At first, I think about change d4 type from int to float, then zeropointone is correct. I think it work out here but it may change the original meaning for the function. So I think in another way, I think about the d4 itself, I can modify it outside the function. So in the miltidivide function, it only divides the first three time, then use the return value to divides float(10). It will not change both function and zeropointone. So it is fixed from **multidivide(f*10, a, a, a, a)** to **multidivide(f*10, a, a, a, 1)/float(a)**

*Why important?*

There will be multiple ways to fix a problem, but you need to find a way you think best. Also, type is very crucial in c++ program, change a kind of type it may change the result.

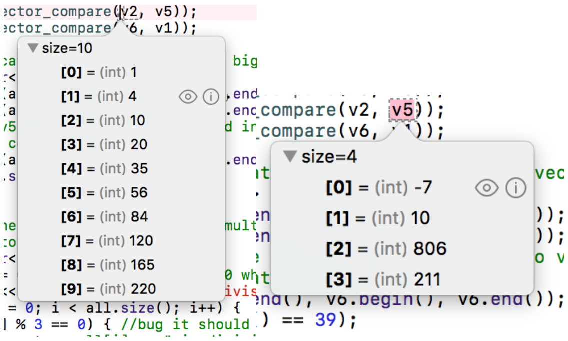
BUG #6 Line 151. In the function vector_compare(v1, v2).

*How to find?*



```
~~Dr.M~~ Error #1: UNADDRESSABLE ACCESS beyond heap bounds: reading 0x005db7a0-0
x005db7a4 4 byte(s)
~~Dr.M~~ # 0 vector_compare                  [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:152]
~~Dr.M~~ # 1 vector_operations               [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:456]
~~Dr.M~~ # 2 main                            [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:633]
~~Dr.M~~ Note: @0:00:01.334 in thread 338813
~~Dr.M~~ Note: refers to 0 byte(s) beyond last valid byte in prior malloc
~~Dr.M~~ Note: prev lower malloc:  0x005db790-0x005db7a0
~~Dr.M~~ Note: instruction: cmp    %eax (%edx,%edi,4)
~~Dr.M~~
~~Dr.M~~ Error #2: UNADDRESSABLE ACCESS beyond heap bounds: reading 0x005db82c-0
x005db830 4 byte(s)
~~Dr.M~~ # 0 vector_compare                  [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:152]
~~Dr.M~~ # 1 vector_operations               [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:457]
~~Dr.M~~ # 2 main                            [/Users/kexin/Library/Mobile Docume
nts/com~apple~CloudDocs/2016 FALL/CSCI 1200/hw4/operations.cpp:633]
~~Dr.M~~ Note: @0:00:01.337 in thread 338813
~~Dr.M~~ Note: refers to 0 byte(s) beyond last valid byte in prior malloc
```

DrMemory print the message said that UNADDRESSABLE ACCESS. It happens twice, and all happen in the line 152. Set the breakpoint around the line 456, and find out what in the two vectors when they compare.



```
ector_compare(v2, v5));
ector_compare(v6, v1));
```

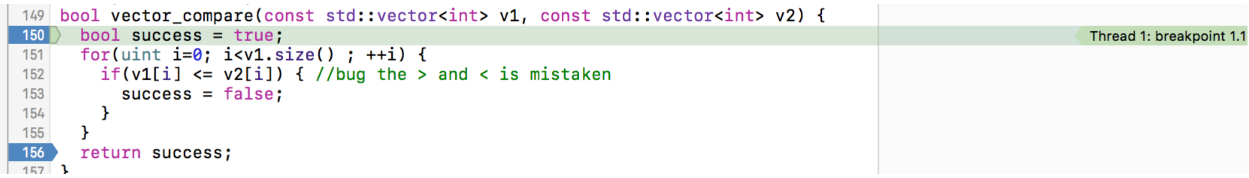
```
size=10
    [0] = (int) 1
    [1] = (int) 4
    [2] = (int) 10
    [3] = (int) 20
    [4] = (int) 35
    [5] = (int) 56
    [6] = (int) 84
    [7] = (int) 120
    [8] = (int) 165
    [9] = (int) 220
```

```
compare(v2, v5));
compare(v6, v1));
```

```
size=4
    [0] = (int) -7
    [1] = (int) 10
    [2] = (int) 806
    [3] = (int) 211
```

```
 = 0; i < all.size(); i++) {
] % 3 == 0) { //bug it should
end(), v6.begin(), v6.end());
) == 39);
```

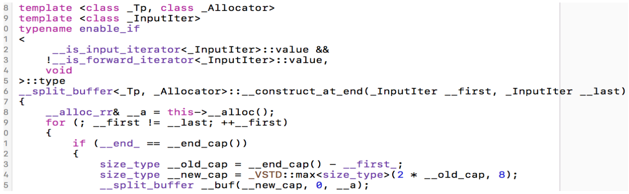
It is easier to see that size of these two vectors are different. Set two breakpoints before and after the for loop.



```
bool vector_compare(const std::vector<int> v1, const std::vector<int> v2) {
  bool success = true;
  for(uint i=0; i<v1.size() ; ++i) {
    if(v1[i] <= v2[i]) { //bug the > and < is mistaken
      success = false;
    }
  }
  return success;
}
```

Thread 1: breakpoint 1.1

It will end up with another several files, such as



```
template <class _Tp, class _Allocator>
template <class _InputIter>
typename enable_if
<
     __is_input_iterator<_InputIter>::value &&
    !__is_forward_iterator<_InputIter>::value,
    void
>::type
__split_buffer<_Tp, _Allocator>::__construct_at_end(_InputIter __first, _InputIter __last)
{
    __alloc_rr& __a = this->__alloc();
    for (; __first != __last; ++__first)
    {
        if (__end_ == __end_cap())
        {
            size_type __old_cap = __end_cap() - __first_;
            size_type __new_cap = _VSTD::max<size_type>(2 * __old_cap, 8);
            __split_buffer __buf(__new_cap, 0, __a);
```

Then I will find out when compare v2 and v5, the index will out of range, since i is larger than 5, the v5 is out of range.

*How to fix?*

Add another conditional in for loop. **i<v1.size()&&v2.size().** This will make sure that when compare the values in two different size vectors, it will never out of range.

*Why important?*

It is important to consider all conditions when we write programs. The buggy program does not think about that two vectors may be different size. This is a common mistake I should pay attention to.