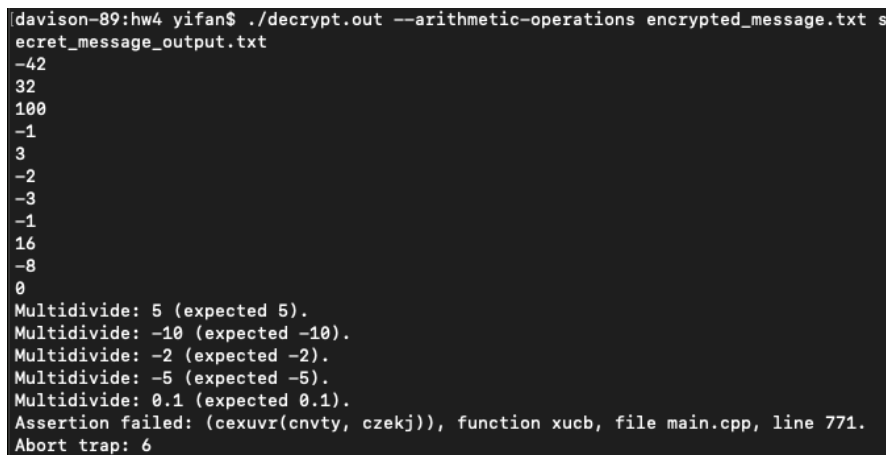


## DS HW4 Writeup

When I first got the code, I read through most of it and tried to fix the most obvious bugs. For example, read the comment in front of each function and check the loop conditions (eg. Wrong start point and end condition in for loop), if conditions (eg. The comparison sign in the function that compare element in two vector; == instead of !=) or the most noticeable logic errors (eg. Using variable before initialize it).

Then, I started to go through operations in the order as the pdf suggested. The first one is arithmetic operation. The compiler shows that the result number is wrong. So my first thinking is that the calculation function's problem. However, even after I cast float in the calculation, the number is still far from what it should be. So it's not the problem of the calculation function, it can only be the problem of setting (assigning)



```
davison-89:hw4 yifan$ ./decrypt.out --arithmetic-operations encrypted_message.txt s
ecret_message_output.txt
-42
32
100
-1
3
-2
-3
-1
16
-8
0
Multidivide: 5 (expected 5).
Multidivide: -10 (expected -10).
Multidivide: -2 (expected -2).
Multidivide: -5 (expected -5).
Multidivide: 0.1 (expected 0.1).
Assertion failed: (cexuvr(cnvty, czekj)), function xucb, file main.cpp, line 771.
Abort trap: 6
```

variables. After I tried to make the variables equal to the numbers in the comment, there was still assertion error like the left image shows. So I used cout to print all the variables and compare

them to the comment numbers. I found that the last variable should be 0.1 instead of 0. That's because both divisor and dividend were int and it will cut off everything after the int part. So I casted float for both of them and make arithmetic operation works correctly. From this, I learned more about int and float arithmetic rules and how to use cout for debugging.

Next one is file operation, most of the errors are just the wrong if conditions such as "argc < 3" which should be 4 and "argc==4" which should be !=. The only part that

has structure error is the part that creating array to hold information like the image shown. The code created dynamic before having the size of the array. So I switched the order of initializing the size variable and creating array which solve the bug.

Then, is the array operation. Except for the part where creates the 2D array has some error with the for loop start/end points and the last index is out of range, the most complicated part is where we tried to print the array with pointer. I got this error shown like this.

So I  
check the  
loop of

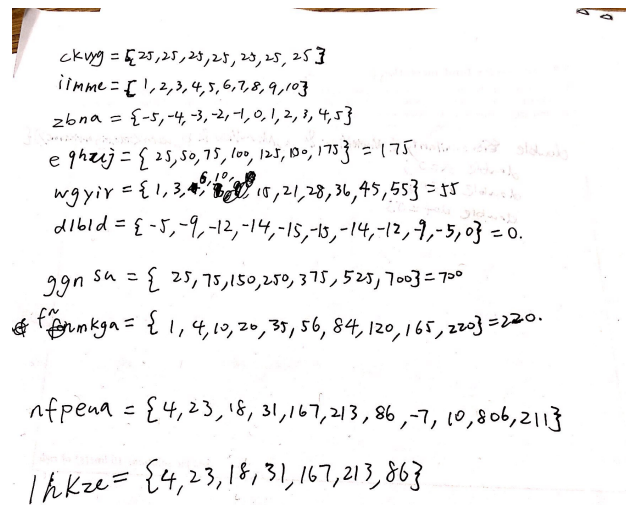
[illegible]

printing again. I found that the second loop was not actually incrementing the pointer in the first loop but the pointer pointing to a 1D array which will cause the outranging of the index and result in segmentation fault. So I changed the pointer incremented in the second loop into the pointer pointing to the individual element that we assigned in the first loop. Then there are just some minor problem with the printing format of the grid. This process helped me review the use of pointer and how to use nested loop to get access to the 2D array.

Furthermore, here's things get troublesome. I first thought the adding up function didn't pass by reference in purpose to just calculate the sum without changing the vector. So, I follow this logic and changed some of the assert functions. However, when I went to the end and test it, it showed error like this graph.

```
[davison-89:hw4 yifan$ g++ main.cpp -lm -o decrypt.out  
[davison-89:hw4 yifan$ ./decrypt.out --vector-operations encrypted_message.txt secret_message_  
output.txt  
Now counting numbers divisible by 3  
There are 8 numbers divisible by 3.  
wxwi[7] = 33  
wxwi[6] = 30  
wxwi[5] = 25  
wxwi[4] = 22  
wxwi[3] = 19  
wxwi[2] = 15  
wxwi[1] = 12  
wxwi[0] = 9  
Segmentation fault: 11
```

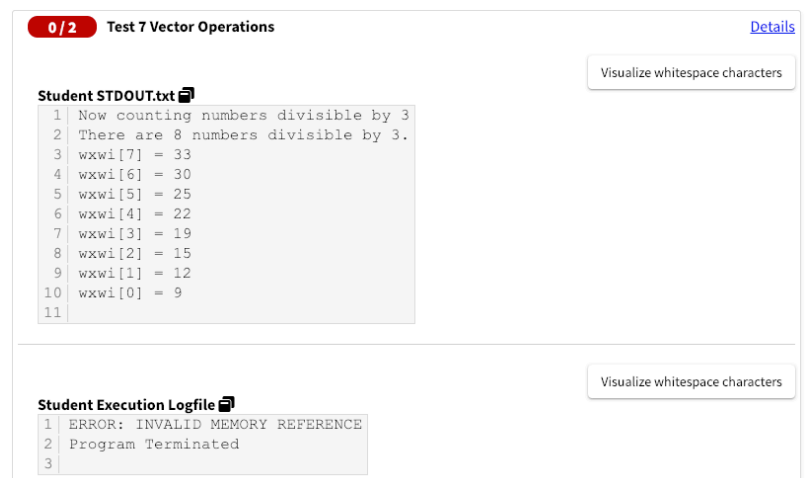
I had no idea what happened inside the vector and where goes wrong. So, I decided to draw out everything in every single vector. The following graph is what I drew that time.



$ckvqg = \{25, 25, 25, 25, 25, 25, 25\}$   
 $ihmmc = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$   
 $zbnna = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$   
 $e9hzej = \{25, 50, 75, 100, 125, 150, 175\} = 175$   
 $wgyir = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55\} = 55$   
 $d1bld = \{-5, -9, -12, -14, -15, -15, -14, -12, -9, -5, 0\} = 0$   
 $ggnsa = \{25, 75, 150, 250, 375, 525, 700\} = 700$   
 $fzmkga = \{1, 4, 10, 20, 35, 56, 84, 120, 165, 220\} = 220$   
 $nfpeaa = \{4, 23, 18, 31, 167, 213, 86, -7, 10, 806, 211\}$   
 $lkze = \{4, 23, 18, 31, 167, 213, 86\}$

I realized that the numbers were totally wrong in my final vector. So, I began to suspect that I should pass by reference and which will also satisfy the original assertions. So I changed the sum function to pass by reference and I got the output I expect and the shown of bug FIXED. But, when I tried to submit it on submittly. It shows a memory error like this.

So, I ran the program again and enter the command -Wall to show all the warning. It showed following warning: “comparison of unsigned expression  $\geq 0$  is always true” on the for loop printing the grid. Because the for loop goes backward, the stop condition is when it reaches 0. However, the type of index was already unsigned int. So, the loop will not stop at that condition. To solve this, I changed the unsigned int into normal int and the problem was fixed. In later programming, we should be caution when using int or unsigned int in a loop so that it will not affect the effectiveness of the loop.



**0/2 Test 7 Vector Operations** [Details](#)

Visualize whitespace characters

**Student STDOUT.txt**

```

1 Now counting numbers divisible by 3
2 There are 8 numbers divisible by 3.
3 wxwi[7] = 33
4 wxwi[6] = 30
5 wxwi[5] = 25
6 wxwi[4] = 22
7 wxwi[3] = 19
8 wxwi[2] = 15
9 wxwi[1] = 12
10 wxwi[0] = 9
11

```

**Student Execution Logfile**

```

1 ERROR: INVALID MEMORY REFERENCE
2 Program Terminated
3

```

Visualize whitespace characters

Last but not the least, the list operation. Most debugging went well and I already printed all the output correctly. However, the terminal still showed segmentation fault which means index out of range. So I used memory debugger. It shows that the memory bug is in the erase inside the mqoms function. Also, the only place the code implemented erase is at the part that “remove every number from the list that is a

multiple of at least one of these pppf”. The original code was “`xh_j_o.erase(xewokj);`”

```
WARNING: this version of Mac OSX is not officially supported by Dr. Memory.
~~Dr.M~~ Dr. Memory version 1.11.0
~~Dr.M~~ WARNING: Dr. Memory for Mac is Beta software. Please report any
~~Dr.M~~ problems encountered to http://drmemory.org/issues.
~~Dr.M~~ WARNING: application is missing line number information.
~~Dr.M~~
~~Dr.M~~ Error #1: UNADDRESSABLE ACCESS of freed memory: reading 4 byte(s)
~~Dr.M~~ # 0 mqoms
~~Dr.M~~ Note: refers to 4 byte(s) into memory that was freed here:
~~Dr.M~~ Note: # 0 replace_operator_delete
~~Dr.M~~ Note: # 1 std::_1::list<>::erase
~~Dr.M~~ Note: # 2 mqoms
~~Dr.M~~ Note: # 3 main
raspberry elderberry nectarine orange zwetschge pomegranate durian grape banana
fig huckleberry strawberry tangerine jujube lemon mango cherry uglyfruit apple
watermelon kiwi
3 letters did not ever appear in the fruit names.
List bugs are FIXED

~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~      1 unique,      110 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~~      0 unique,       0 total,       0 byte(s) of leak(s)
~~Dr.M~~      0 unique,       0 total,       0 byte(s) of possible leak(s)
```

this code erase the element pointed by the iterator `xewokj`. However, after erase it, the iterator still point to that but not in the original list. We need to use the statement “`xewokj = xh_j_o.erase(xewokj);`” to make the iterator points to the next element after erasing the previous one. At the same time, since the for loop already

increment the iterator every time. We need to decrement the iterator manually after erase. Finally, the memory bug was fixed. This helped me review the knowledge of list erasing technique that erase may cause invalidation of the iterator we should be very careful doing this kind of manipulation.