ECE 2035 In Class Activity:
<u>Detecting and Debugging Errors in Dynamically Allocated Structures</u>

Objective: Creating and manipulating dynamically allocated data structures, such as lists and hash tables, on the heap can be error-prone. This is particularly difficult in multi-file systems which contain code not completely written by a single programmer (e.g., the common situation in which you need to collaborate with others, using code from multiple libraries or programmers). The objective of this exercise is to give you practice with debugging strategies and tools by giving you a program that has common memory errors and errors in the composition of code from multiple files. For this exercise, you will debug the provided program in C to have it function correctly.

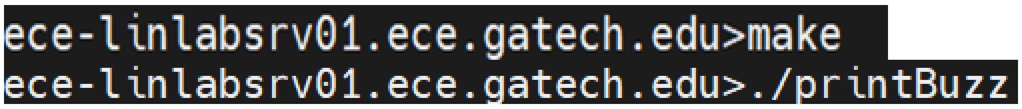
Honor Policy: For this in class activity, you are encouraged to work with your peers around you, but you must submit your own corrections to the code, documentation of them in the code, and the completion of the report section (below).

Your corrections must be documented in the report section for full credit.

Debugging Assignment: A long time ago, George Burdell was enrolled in ECE 2023 here at Georgia Tech. He had an assignment in which he had to read in a data stream of integer values from an encrypted file, decrypt each integer, store it in a singly linked list, and then finally print it to his display. George has coded up the assignment per his instructions, but he can't get Buzz to print. Can you help him debug his program to print Buzz?

The provided files given to you include main.c, graphics.c, security.c, ll.c and their associated header files. It is your job to find the bugs and correct the code as well as to document them in a txt file called bug.txt. ***There will be no bugs in the header files and do not change the header files.*** If you need to change a header file to fix a bug, you are on the wrong track. Unless specified in a comment, assume that any function may have a bug.

You should compile and run your program on the ECE Linux Lab machines (https://help.ece.gatech.edu/labs/names), using the following commands:



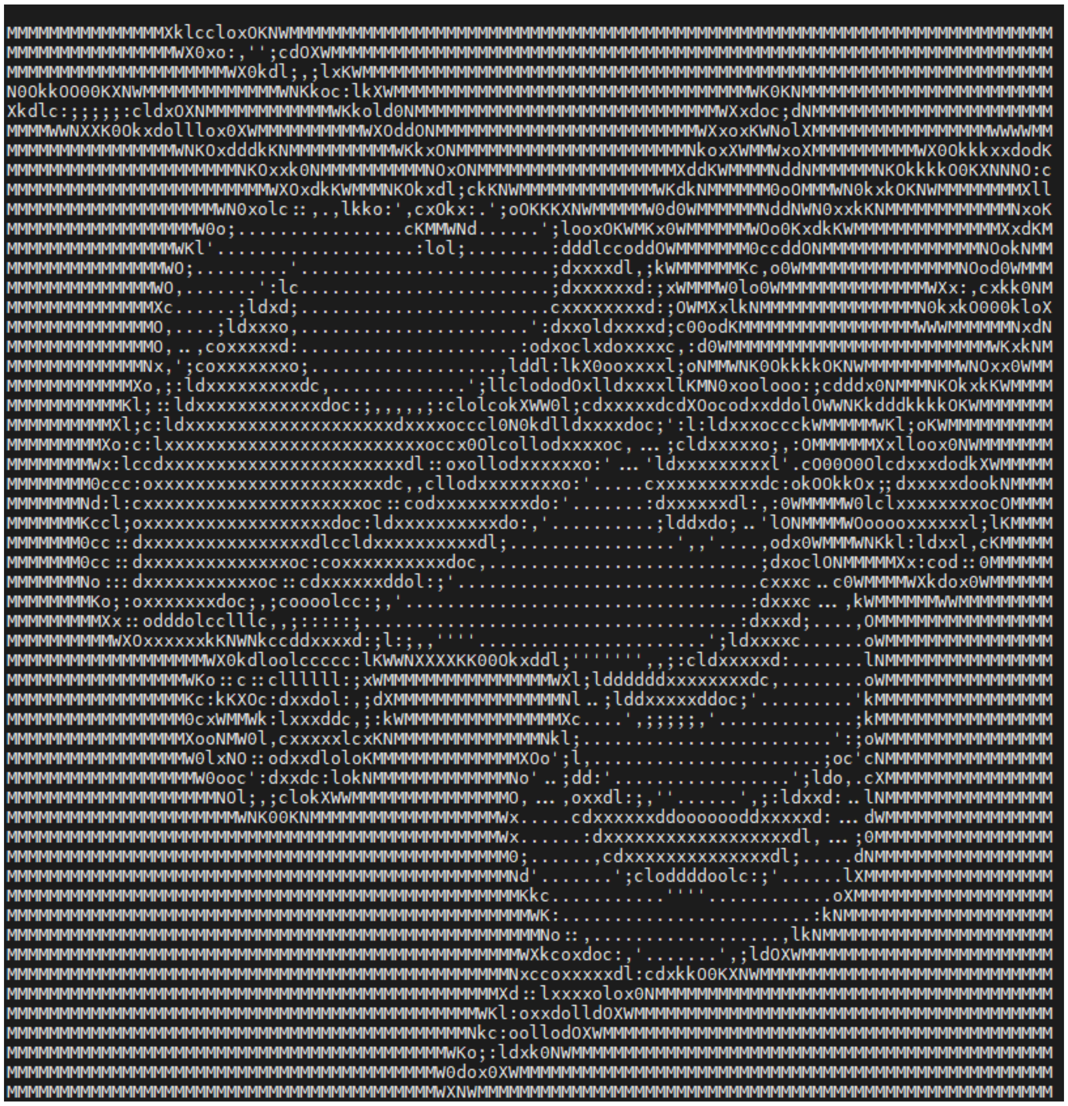
```
ece-linlabsrv01.ece.gatech.edu>make
ece-linlabsrv01.ece.gatech.edu>./printBuzz
```

Any errors you see when compiling code may be a hint as to where some bugs are as well.

We are not looking for any fixes that may speed up the program/make it more efficient. Just bugs that are prevent Buzz from being printed without any error messages.

ECE 2035 In Class Activity:
<u>Detecting and Debugging Errors in Dynamically Allocated Structures</u>

You will know you have found all the bugs when you see Buzz printed, such as in the image below:

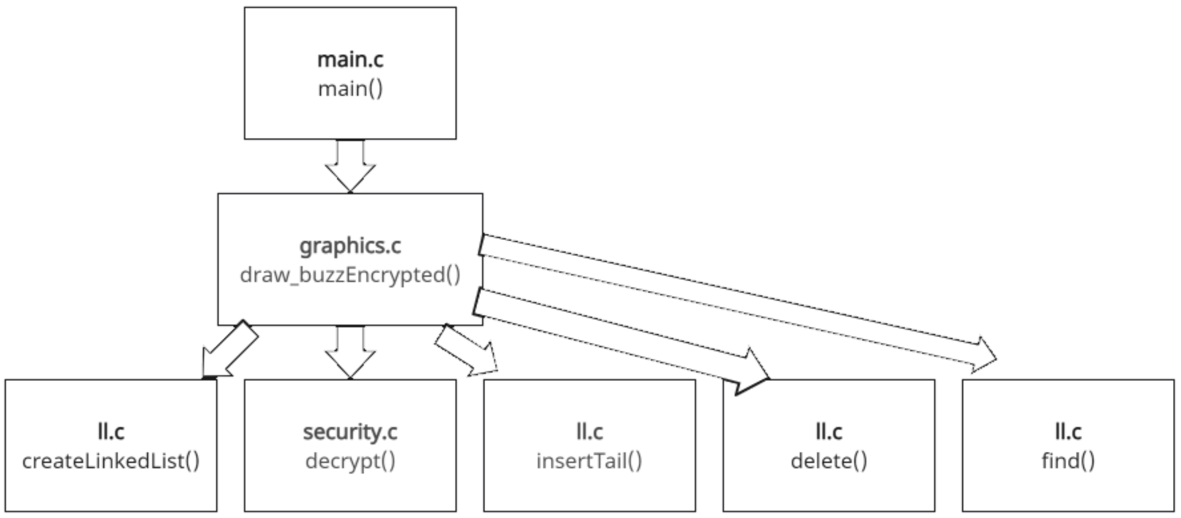


```
MMMMMMMMMMMMMXklcclox0KNWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWX0xo:,'';cd0XWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMWX0kdl;,;lxKWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
N0OkkOOO0KXNWMMMMMMMMMMMMMWNKkoc:lkXWMMMMMMMMMMMMMMMMMMMMMWK0KNMMMMMMMMMMMMMMMMMMMMMMMMMM
Xkdlc:;;;;;;:cldxOXNMMMMMMMMMMMMWKkold0NMMMMMMMMMMMMMMMMMMMWXxdoc;dNMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMWWNXXK0OkxdollloxOXWMMMMMMMMMMWXOdd0NMMMMMMMMMMMMMMMMMMMMMWXxoxKWNolXMMMMMMMMMMMMMMMWWWWMM
MMMMMMMMMMMMMMMWNKOxdddkKNMMMMMMMMMWKkxONMMMMMMMMMMMMMMMMMMNkoxXWMMWxoXMMMMMMMMMWX0OkkkxxdodK
MMMMMMMMMMMMMMMMMMMMMMNKOxxk0NMMMMMMMMMMN0xONMMMMMMMMMMMMMMMMMXddKWMMMMNddNMMMMMMNKOkkkkO0KXNNNO:c
MMMMMMMMMMMMMMMMMMMMMMMMMMWXOxdkKWMMMNKOkxdl;ckKNWMMMMMMMMMMMMMMWKdkNMMMMMM0oOMMMWN0kxkOKNWMMMMMMMMXll
MMMMMMMMMMMMMMMMMMMMMWN0xolc::,.,lkko:',cxOkx:.';oOKKKXNWMMMMMW0d0WMMMMMMNddNWN0xxkKNMMMMMMMMMMMMMNxoK
MMMMMMMMMMMMMMMMMMW0o;...............cKMMWNd......';looxOKWMKx0WMMMMMMWOo0KxdkKWMMMMMMMMMMMMMMMXxdKM
MMMMMMMMMMMMMMMMMWKl'....................:lol;........:dddlccoddOWMMMMMMM0ccddONMMMMMMMMMMMMMMMMNOokNMM
MMMMMMMMMMMMMMMWO;........'.......................;dxxxxdl,;kWMMMMMMKc,o0WMMMMMMMMMMMMMMMMNOod0WMMM
MMMMMMMMMMMMMMWO,.......':lc......................;dxxxxxxd:;xWMMMW0lo0WMMMMMMMMMMMMMMWXx:,cxkk0NM
MMMMMMMMMMMMMXc......;ldxd;.......................cxxxxxxxxd:;OWMXxlkNMMMMMMMMMMMMMMMN0kxkO000kloX
MMMMMMMMMMMMMO,....;ldxxxo,.....................':dxxoldxxxxd;c00odKMMMMMMMMMMMMMMMMMWWWMMMMMMMNxdN
MMMMMMMMMMMMMO,..,coxxxxxd:....................:odxoclxdoxxxxc,:d0WMMMMMMMMMMMMMMMMMMMMMMMMMWKxkNM
MMMMMMMMMMMMNx,';coxxxxxxxxo;..................,lddl:lkX0ooxxxxl;oNMMWNK0OkkkkOKNWMMMMMMMMMMWNOxx0WMM
MMMMMMMMMMMXo,;:ldxxxxxxxxxdc,............';llclododOxlldxxxxllKMN0xoolooo:;cdddx0NMMMMNKOkxkKWMMMM
MMMMMMMMMMKl;::ldxxxxxxxxxxxxdoc:;,,,,,,;:clolcokXWW0l;cdxxxxxdcdXOocodxxddolOWWNKkdddkkkkOKWMMMMMMM
MMMMMMMMMXl;c:ldxxxxxxxxxxxxxxxxxxxxdxxxxoccclON0kdlldxxxxdoc;':l:ldxxxoccckWMMMMMWKl;oKWMMMMMMMMMM
MMMMMMMMXo:c:lxxxxxxxxxxxxxxxxxxxxxxxxoccx0Olcollodxxxxoc,…;cldxxxxxo;,:OMMMMMMXxlloox0NWMMMMMMMM
MMMMMMMWx:lccdxxxxxxxxxxxxxxxxxxxxxxxxdl::oxollodxxxxxxo:'…'ldxxxxxxxxxl'.cO00O0lcdxxxdodkXWMMMMM
MMMMMMM0ccc:oxxxxxxxxxxxxxxxxxxxxxxxdc,,cllodxxxxxxxxo:'.....cxxxxxxxxxxdc:okOOkkOx;;dxxxxxdookNMMMM
MMMMMMNd:l:cxxxxxxxxxxxxxxxxxxxxxxoc::codxxxxxxxxxxdo:'.......:dxxxxxxdl:,:0WMMMMW0lclxxxxxxxxocOMMMM
MMMMMMKccl;oxxxxxxxxxxxxxxxxxxxdoc:ldxxxxxxxxxxdo:,'..........;lddxdo;..'lONMMMMWOooooxxxxxxl;lKMMMM
MMMMMM0cc::dxxxxxxxxxxxxxxxxdlccldxxxxxxxxxxxdl;................',,'....,odx0WMMMWNKkl:ldxxl,cKMMMMM
MMMMMM0cc::dxxxxxxxxxxxxxxxoc:coxxxxxxxxxxxdoc,.........................;dxoclONMMMMMXx:cod::0MMMMMM
MMMMMMNo:::dxxxxxxxxxxxxoc::cdxxxxxxddol:;'.............................cxxxc..c0WMMMMWXkdox0WMMMMMM
MMMMMMMKo;:oxxxxxxxxdoc;,;coooolcc:;,'..................................:dxxxc…,kWMMMMMMWWMMMMMMMMM
MMMMMMMMXx::odddolcclllc,,;:::::;....................................:dxxxd;....,OMMMMMMMMMMMMMMMMM
MMMMMMMMMWXOxxxxxxkKNWNkccddxxxxd:;l:;,,''''.....................';ldxxxxc......oWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMWX0kdloolccccc:lKWWNXXXXKK00Okxddl;'''''',,;:cldxxxxxd:.......lNMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMWKo::c::clllllll:;xWMMMMMMMMMMMMMMMMMWXl;ldddddxxxxxxxxdc,........oWMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMKc:kKXOc:dxxdol:,;dXMMMMMMMMMMMMMMMMMNl..;lddxxxxxddoc;'.........'kMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMM0cxWMMWk:lxxxddc,;:kWMMMMMMMMMMMMMMMMXc....',;;;;;,'............;kMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMXooNMW0l,cxxxxxlcxKNMMMMMMMMMMMMMMMMNkl;.........................':;oWMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMW0lxNO::odxxdloloKMMMMMMMMMMMMMMMMXOo';l,......................;oc'cNMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMW0ooc':dxxdc:lokNMMMMMMMMMMMMMMMNo'..;dd:'.................';ldo,.cXMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMNOl;,;clokXWWMMMMMMMMMMMMMMMMMO,…,oxxdl:;,''......',;:ldxxd:..lNMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMWNK00KNMMMMMMMMMMMMMMMMMMMMMMWx.....cdxxxxxxddoooooddxxxxxd:…dWMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWx......:dxxxxxxxxxxxxxxxxxxxdl,…;0MMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM0;......,cdxxxxxxxxxxxxxxxdl;.....dNMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMNd'.......';cloddddoolc:;'......lXMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMKkc..........''''...........oXMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWK:......................:kNMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMNo::,..................,lkNMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWXkcoxdoc:,'.......',;ldOXWMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMNxccoxxxxxdl:cdxkkO0KXNWMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMXd::lxxxxolox0NMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKl:oxxdolldOXWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMNkc:oollodOXWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWKo;:ldxk0NWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMW0dox0XWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWXNWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
```

In the corrected code that you hand in, there should be no additional output, such as print statements that you may have added to help debug, or any errors still present in the program. This includes errors from the address sanitizer. You should see this **exact image above**, if you see Buzz but he doesn't look like the **black and white** image above, you probably still have an unfound bug.

ECE 2035 In Class Activity:
<u>Detecting and Debugging Errors in Dynamically Allocated Structures</u>

Call Graph: The following figure shows what functions from which files are called in what order. This may help you determine where bugs may be in the assignment.

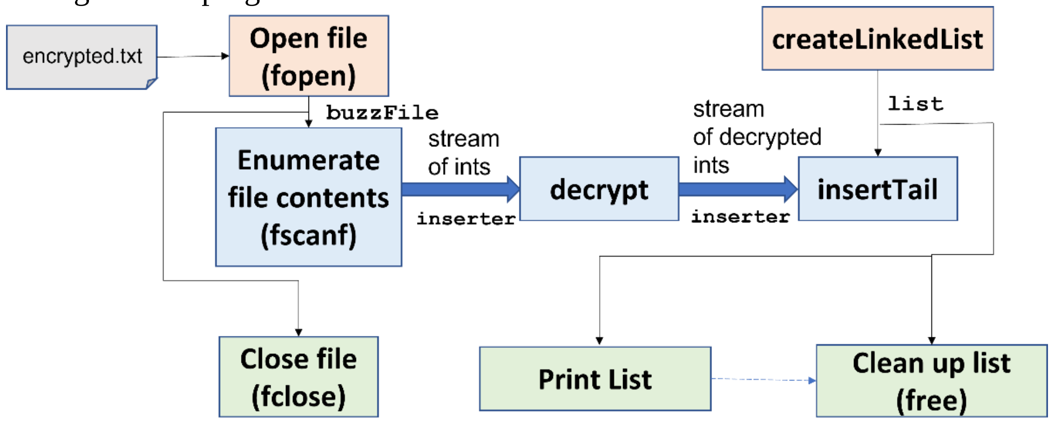


You may notice that encrypt() from security.c is not called by this program. Rather, it has been used earlier by a different program to encrypt the original image file to generate the given file encrypted.txt. The encrypt() function is correct code, showing how the original file was encrypted. It is intended to show correct code and how the file is encrypted.

When doing this activity, please do the tasks in order. If you think you see a bug in a file, it will be likely be addressed as you work through the activity. Think of it as a guided scavenger hunt in code.

If you get stuck or confused, don't hesitate to ask your peers around you, or raise your hand for an instructor to come assist and get you back on the right track.

Dataflow Graph: The following figure shows how the data read from the file is manipulated throughout the program



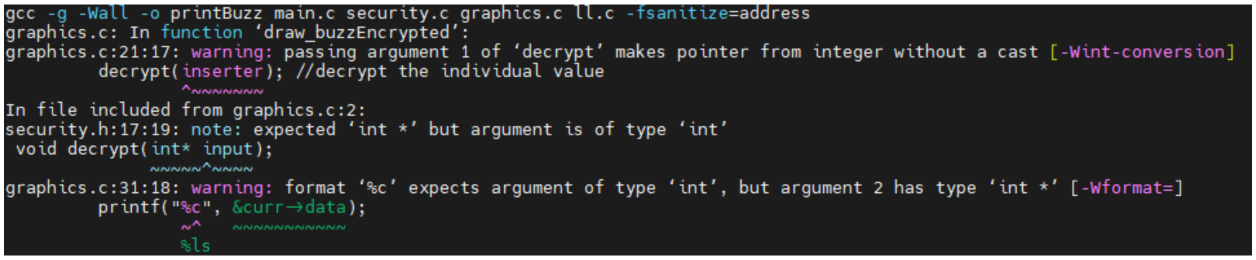
ECE 2035 In Class Activity:
Detecting and Debugging Errors in Dynamically Allocated Structures

Grading: To receive full credit, you must comment in the program where you find bugs, fix the code, and then fill out the report section.

Report:

Task1:

When you first run make you will see warnings caught already by the compiler. Let us investigate these warnings first. (Note, all screenshots may not have the same line numbers as your file, go to the appropriate line number in your warnings)



```
gcc -g -Wall -o printBuzz main.c security.c graphics.c ll.c -fsanitize=address
graphics.c: In function ‘draw_buzzEncrypted’:
graphics.c:21:17: warning: passing argument 1 of ‘decrypt’ makes pointer from integer without a cast [-Wint-conversion]
        decrypt(inserter); //decrypt the individual value
                ^~~~~~~~
In file included from graphics.c:2:
security.h:17:19: note: expected ‘int *’ but argument is of type ‘int’
 void decrypt(int* input);
              ~~~~~^~~~~
graphics.c:31:18: warning: format ‘%c’ expects argument of type ‘int’, but argument 2 has type ‘int *’ [-Wformat=]
        printf("%c", &curr->data);
                 ~^   ~~~~~~~~~~~
                 %ls
```

The first warning says that decrypt accepts a pointer instead of an integer. If we investigate that line in graphics.c we see decrypt is being called with integer inserter. Now if we investigate the definition of decrypt in security.h we see that it wants a pointer to be passed in.

Describe the fix you implemented here to get rid of the warning and why it works:

- line 21: inserter &inserter

It works because we need to input an andress to the integer and not pass by value

Now we can move onto the second warning. This warning says that “%c” expects an integer to be given to in after the comma, but it says that the argument has type int *.

Describe the fix you implemented here to get rid of the warning and why it works:

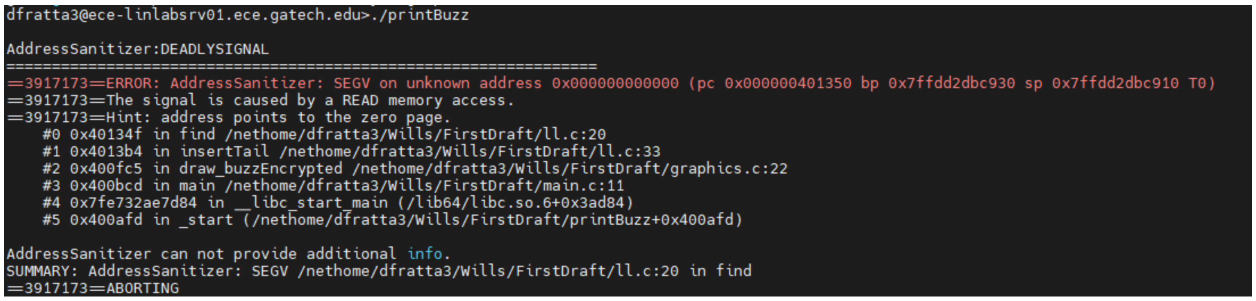
- line 31: &curr-> data curr-> data

curr is already a pointer and so we do not need the address of curr

Task 2:

After fixing these warnings, save your code and re-run make. Here you will see that there are no warnings and it is a good time to run the executable. Running the executable is as simple as typing “./printBuzz” on the command line you are using.

ECE 2035 In Class Activity:
Detecting and Debugging Errors in Dynamically Allocated Structures



```
dfratta3@ece-linlabsrv01.ece.gatech.edu>./printBuzz

AddressSanitizer:DEADLYSIGNAL
=================================================================
==3917173==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x000000401350 bp 0x7ffdd2dbc930 sp 0x7ffdd2dbc910 T0)
==3917173==The signal is caused by a READ memory access.
==3917173==Hint: address points to the zero page.
    #0 0x40134f in find /nethome/dfratta3/Wills/FirstDraft/ll.c:20
    #1 0x4013b4 in insertTail /nethome/dfratta3/Wills/FirstDraft/ll.c:33
    #2 0x400fc5 in draw_buzzEncrypted /nethome/dfratta3/Wills/FirstDraft/graphics.c:22
    #3 0x400bcd in main /nethome/dfratta3/Wills/FirstDraft/main.c:11
    #4 0x7fe732ae7d84 in __libc_start_main (/lib64/libc.so.6+0x3ad84)
    #5 0x400afd in _start (/nethome/dfratta3/Wills/FirstDraft/printBuzz+0x400afd)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /nethome/dfratta3/Wills/FirstDraft/ll.c:20 in find
==3917173==ABORTING
```

Oh no! We have encountered a segmentation violation. This means that our program is trying to access memory it does not have permission to use. When you see a segmentation violation, it is important to start investigating code that affects memory, things such as your mallocs, free, and pointer logic. Here we see that the source of the error is in the find method in ll.c. Note that the SUMMARY produced by the AddressSanitizer mentions the most likely location of the bug.

Open the find method in ll.c, investigate the method. Identify what is causing a segmentation violation and explain your fix and why it works:

```
while(this_node)
```

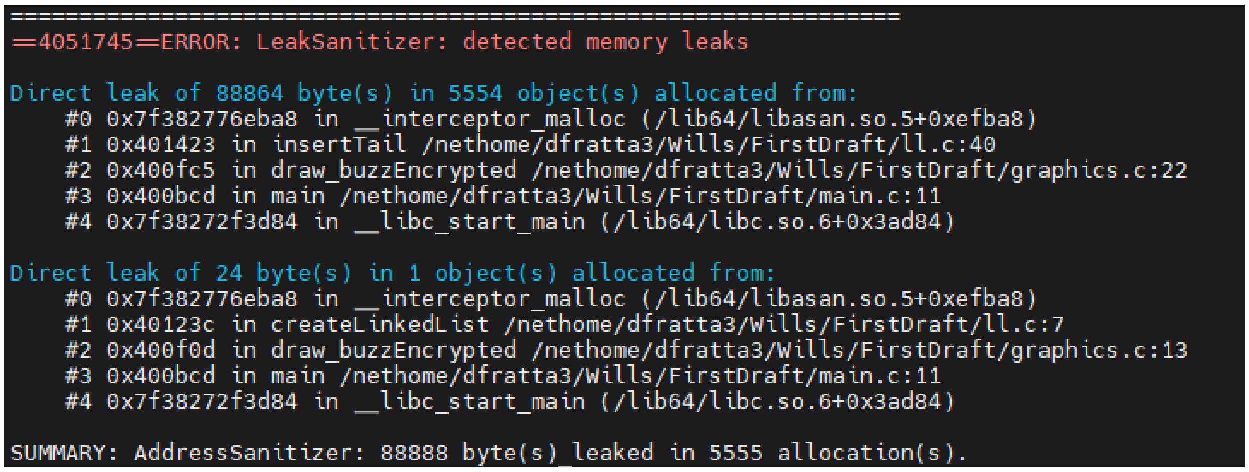
while (1)

while (this_node)

This makes sure that the while loop exits when this_node is null.

Task 3:

Once you find and fix the segmentation violation, save your code, run make, and run the executable again.



```
=================================================================
==4051745==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 88864 byte(s) in 5554 object(s) allocated from:
    #0 0x7f382776eba8 in __interceptor_malloc (/lib64/libasan.so.5+0xefba8)
    #1 0x401423 in insertTail /nethome/dfratta3/Wills/FirstDraft/ll.c:40
    #2 0x400fc5 in draw_buzzEncrypted /nethome/dfratta3/Wills/FirstDraft/graphics.c:22
    #3 0x400bcd in main /nethome/dfratta3/Wills/FirstDraft/main.c:11
    #4 0x7f38272f3d84 in __libc_start_main (/lib64/libc.so.6+0x3ad84)

Direct leak of 24 byte(s) in 1 object(s) allocated from:
    #0 0x7f382776eba8 in __interceptor_malloc (/lib64/libasan.so.5+0xefba8)
    #1 0x40123c in createLinkedList /nethome/dfratta3/Wills/FirstDraft/ll.c:7
    #2 0x400f0d in draw_buzzEncrypted /nethome/dfratta3/Wills/FirstDraft/graphics.c:13
    #3 0x400bcd in main /nethome/dfratta3/Wills/FirstDraft/main.c:11
    #4 0x7f38272f3d84 in __libc_start_main (/lib64/libc.so.6+0x3ad84)

SUMMARY: AddressSanitizer: 88888 byte(s) leaked in 5555 allocation(s).
```

ECE 2035 In Class Activity:
<u>Detecting and Debugging Errors in Dynamically Allocated Structures</u>

Now you see that there is a memory leak error in your code. A memory leak means that there is memory that was allocated but never freed. The error says that the error is coming from a malloc call inside of insertTail in ll.c.

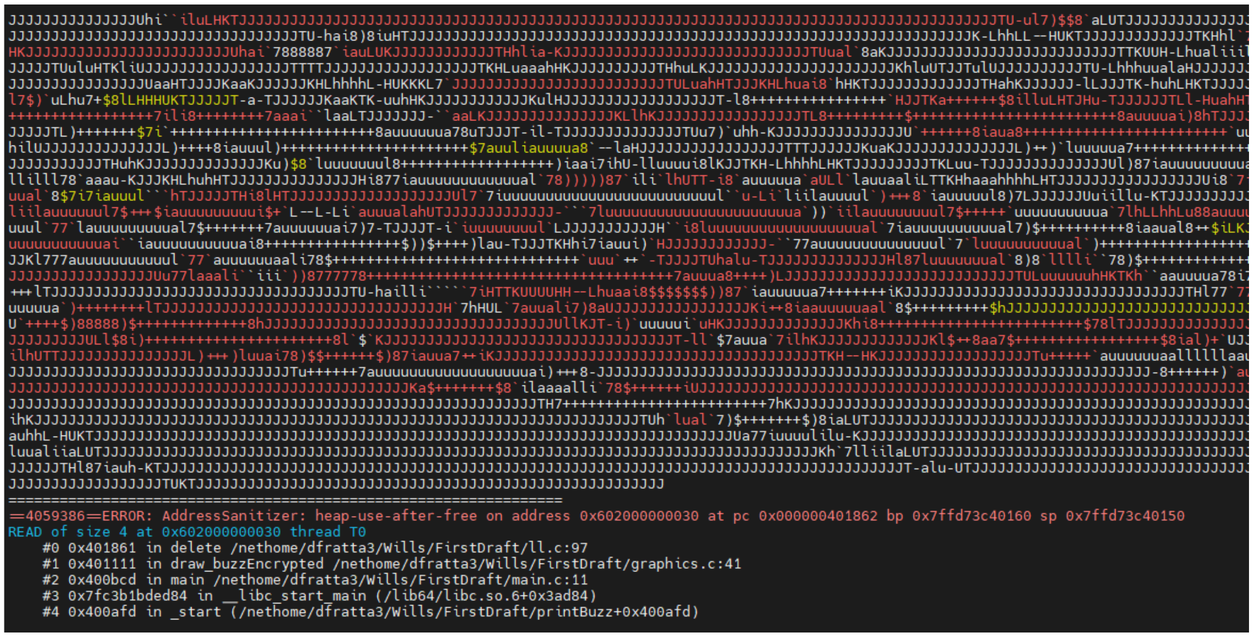
Open up insertTail in ll.c and investigate what may be causing this issue and any other issues. Remember that insertTail needs to be inserting new nodes at the end of the linkedList. Explain the fix you implemented and why it resolves the issue.

```
lList->tail = this_node;
```

Add this in the null case and the else case so that the tail pointer points to the tail and inserting a node.

Task 4:

After fixing insertTail, recompile and run the executable again, you should now see a new error.



```
JJJJJJJJJJJJJJJUhi``iluLHKTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTU-ul7)$$8`aLUTJJJJJJJJJJJJJJJ
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTU-hai8)8iuHTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJK-LhhLL--HUKTJJJJJJJJJJJJJJTKHhl`
HKJJJJJJJJJJJJJJJJJJJJJJJJJUhai`7888887`iauLUKJJJJJJJJJJJJTHhlia-KJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTUual`8aKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTTKUUH-Lhualiiil
JJJJJTUuluHTKliUJJJJJJJJJJJJJJJJJJTTTTJJJJJJJJJJJJJJJJJJTKHLuaaahHKJJJJJJJJJJTHhuLKJJJJJJJJJJJJJJJJJJJJJJJJKhluUTJJTulUJJJJJJJJJJJTU-LhhhuualaHJJJJJJ
JJJJJJJJJJJJJJJJUaaHTJJJJKaaKJJJJJJKHLhhhhL-HUKKKL7`JJJJJJJJJJJJJJJJJJJJJJJJJJJJTULuahHTJJJKHLhuai8`hHKTJJJJJJJJJJJJJJTHahKJJJJJJ-lLJJJTK-huhLHKTJJJJ
l7$)`uLhu7+$8lLHHHUKTJJJJJT-a-TJJJJJJKaaKTK-uuhHKJJJJJJJJJJJJKulHJJJJJJJJJJJJJJJJJT-l8+++++++++++++++`HJJTKa++++++$8illuLHTJHu-TJJJJJJTLl-HuahHT
+++++++++++++++++7ili8++++++++7aaai``laaLTJJJJJJJ-``aaLKJJJJJJJJJJJJJJJJJKLlhKJJJJJJJJJJJJJJJJJJTL8+++++++++$+++++++++++++++++++++++++8auuuuai)8hTJJJ
JJJJJTL)+++++++$7i`+++++++++++++++++++++++8auuuuuua78uTJJJT-il-TJJJJJJJJJJJJJJJTUu7)`uhh-KJJJJJJJJJJJJJJJJU`++++++8iaua8+++++++++++++++++++++++++++`uu
hilUJJJJJJJJJJJJJJL)++++8iauuul)++++++++++++++++++++++$7auuliauuuua8`--laHJJJJJJJJJJJJJJJJJJJTTTJJJJJJKuaKJJJJJJJJJJJJJJJL)++)`luuuuua7+++++++++++++
JJJJJJJJJJJTHuhKJJJJJJJJJJJJJJKu)$8`luuuuuuul8+++++++++++++++++)iaai7ihU-lluuuui8lKJJTKH-LhhhhLHKTJJJJJJJJJTKLuu-TJJJJJJJJJJJJJJJUl)87iauuuuuuuua
llilll78`aaau-KJJJKHLhuhHTJJJJJJJJJJJJJJJHi877iauuuuuuuuuuuual`78)))))87`ili`lhUTT-i8`auuuuua`aULl`lauuaaliLTTKHhaaahhhhLHTJJJJJJJJJJJJJJJJJJJUi8`7
uual`8$7i7iauuul```hTJJJJJTHi8lHTJJJJJJJJJJJJJJJJJJJJJUl7`7iuuuuuuuuuuuuuuuuuuuuuuuuuuul``u-Li`liilauuuul`)+++8`iauuuuul8)7LJJJJJJUuiillu-KTJJJJJJJJJ
liilauuuuuul7$+++$iauuuuuuuuuui$+`L--L-Li`auuualahUTJJJJJJJJJJJJJ-```7luuuuuuuuuuuuuuuuuuuuuua`))`iilauuuuuuuuul7$+++++`uuuuuuuuuua`7lhLLhhLu88auuuu
uuul`77`lauuuuuuuuual7$+++++++7auuuuuuai7)7-TJJJJT-i`iuuuuuuuul`LJJJJJJJJJJJH``i8luuuuuuuuuuuuuuuuuuuual`7iauuuuuuuuuual7)$++++++++++8iaaual8++$iLKJ
uuuuuuuuuuuai``iauuuuuuuuuuai8+++++++++++++++$))$++++)lau-TJJJTKHhi7iauui)`HJJJJJJJJJJJ-``77auuuuuuuuuuuuuuul`7`luuuuuuuuuual`)+++++++++++++++++
JJKl777auuuuuuuuuuuul`77`auuuuuuaali78$++++++++++++++++++++++++++++++`uuu`++`-TJJJJTUhalu-TJJJJJJJJJJJJJJHl87luuuuuuual`8)8`lllli``78)$+++++++++++
JJJJJJJJJJJJJJJJJUu77laaali``iii`))8777778+++++++++++++++++++++++++++++++++++++++++++7auuua8++++)LJJJJJJJJJJJJJJJJJJJJJJJJJJJJTULuuuuuuhHKTKh``aauuuua78i7
+++lTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTU-hailli`````7iHTTKUUUUHH--Lhuaai8$$$$$$$))87`iauuuuua7+++++++iKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTHl77`77
uuuuua`)++++++++lTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJH`7hHUL`7auuali7)8aUJJJJJJJJJJJJJJJJJKi++8iaauuuuuaal`8$+++++++++$hJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
U`++++$)88888)$+++++++++++++8hJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJUllKJT-i)`uuuuui`uHKJJJJJJJJJJJJJJJKhi8+++++++++++++++++++++++++$78lTJJJJJJJJJJJJJJ
JJJJJJJJJJULl$8i)+++++++++++++++++++++8l`$`KJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJT-ll`$7auua`7ilhKJJJJJJJJJJJJJJKl$++8aa7$++++++++++++++++$8ial)+`UJJ
ilhUTTJJJJJJJJJJJJJJJJL)+++)luuai78)$$++++++$)87iauua7++iKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTKH--HKJJJJJJJJJJJJJJJJJJJJJTu+++++`auuuuuuaalllllllaau
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTu++++++7auuuuuuuuuuuuuuuuuuai)+++8-JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ-8++++++)`au
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJKa$+++++++$8`ilaaaalli`78$++++++iUJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTH7++++++++++++++++++++++++7hKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
ihKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJTUh`lual`7)$+++++++$)8iaLUTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
auhhL-HUKTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJUa77iuuuulilu-KJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
luualiiaLUTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJKh`7lliilaLUTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJJTHl87iauh-KTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJT-alu-UTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
JJJJJJJJJJJJJJJJJJTUKTJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
=================================================================
==4059386==ERROR: AddressSanitizer: heap-use-after-free on address 0x602000000030 at pc 0x000000401862 bp 0x7ffd73c40160 sp 0x7ffd73c40150
READ of size 4 at 0x602000000030 thread T0
    #0 0x401861 in delete /nethome/dfratta3/Wills/FirstDraft/ll.c:97
    #1 0x401111 in draw_buzzEncrypted /nethome/dfratta3/Wills/FirstDraft/graphics.c:41
    #2 0x400bcd in main /nethome/dfratta3/Wills/FirstDraft/main.c:11
    #3 0x7fc3b1bded84 in __libc_start_main (/lib64/libc.so.6+0x3ad84)
    #4 0x400afd in _start (/nethome/dfratta3/Wills/FirstDraft/printBuzz+0x400afd)
```

For now, ignore the large block of characters, we will get to that later. Now there is a heap-use-after-free error in the delete method in ll.c

Open delete in ll.c and investigate what may be causing this issue. In this program, we are deleting from the head. What does this mean we have to do about keeping the structure of the linked list maintained while we delete from the head? Identify the issue and explain your fix:

```
lList->head = tmp;
```

Make the head pointer point to tmp after deleting.

ECE 2035 In Class Activity: Detecting and Debugging Errors in Dynamically Allocated Structures

Task 5:

Now after you have fixed the delete method, recompile and run the executable. You should see no errors/warnings, but buzz is not being printed correctly. The large block of characters that makes no sense should be there. Something must be going wrong when decrypting the file, because we know it is being encrypted correctly. Look at both encrypt and decrypt in security.c.

Identify the issue in decrypt and explain the fix you did:

```
*input -= 3;
*input /= 20;
```

Based off of the encrypt strategy, the right way to decrypt is first -3 and then divide by 20.

After you fix the issue in decrypt, recompile and run the program. You should now successfully see Buzz printed on the screen with no warnings or errors.

Task 6:

Lastly, replace the call of insertTail in draw_buzzEncrypted with a call to insertHead, it should still take the same parameters in the same order as insertTail.

What do you see printed? Explain why you see the image you see with the context of a linked list.

We see Inverted buzz. This is because the order in which we are printing buzz has the first go in last.

Congratulations, you are done with the in class debugging activity! ZIP up the files you edited and submit that along with this document after renaming it to

<first name>_<last_name>_<original file name>.pdf