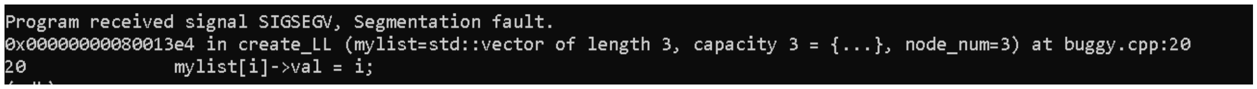
PA 0: Environment Setup, AddressSanitizer and GDB

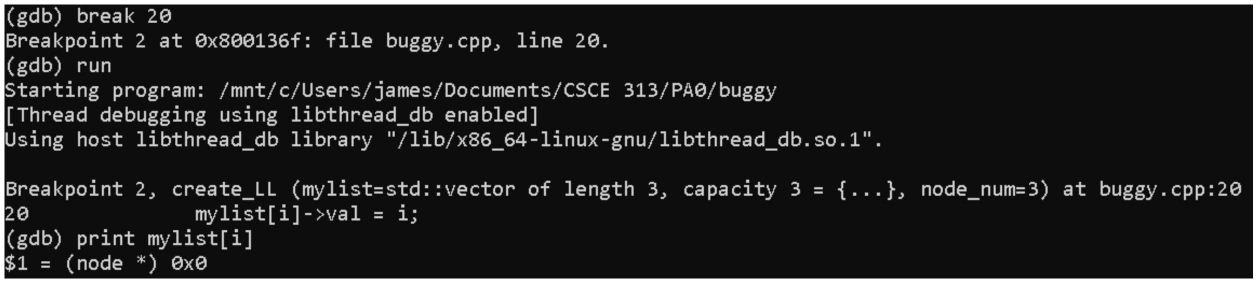
James Robinson

1. **[System Setup]** I set up a virtual machine using Hyper-V and Ubuntu 20.04. I then installed the necessary packages and softwares.
2. **[C++ Compilation]** I used the terminal and g++ to compile buggy.cpp. I got several errors, all of them in the areas where the 'blanks' were'
    a. I included the vector file as well as the 'using namespace std;' statement.
    b. I used the 'public' keyword to make it so that the member variables could be accessed outside the class.
    c. I then changed some of the dot (.) operators to arrow (->) operators so that the pointer's member variables could be accessed.
3. **[Compilation with Symbol Table]** The program then compiled successfully. I added the '-g' tag so that the symbols would show up in the gdb debugger.
4. **[GDB Start/Run/Backtrace]** I ran the program and received this error:



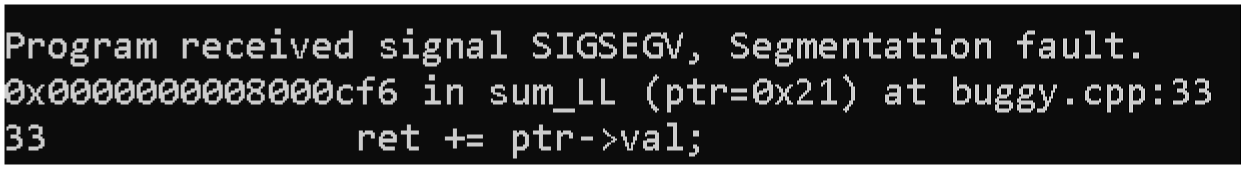
```
Program received signal SIGSEGV, Segmentation fault.
0x00000000080013e4 in create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=3) at buggy.cpp:20
20              mylist[i]->val = i;
```

5. **[GDB Breakpoint/Print]** I put a breakpoint on the line where the segmentation fault occurred and found that 'mylist[i]' was a null pointer.



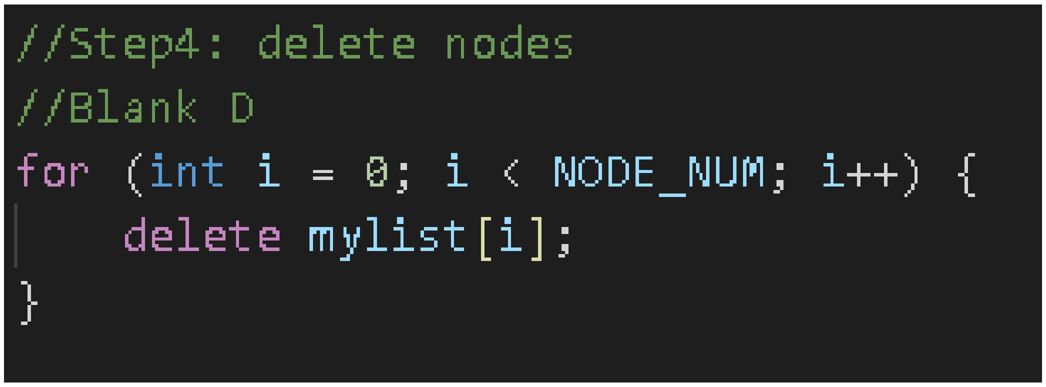
```
(gdb) break 20
Breakpoint 2 at 0x800136f: file buggy.cpp, line 20.
(gdb) run
Starting program: /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 2, create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=3) at buggy.cpp:20
20              mylist[i]->val = i;
(gdb) print mylist[i]
$1 = (node *) 0x0
```

6. **[C++ Runtime Error Fix (Null-Pointer)]** I fixed this error by writing the statement 'mylist[i] = new node;' before assigning the values.
7. **[C++ Runtime Error Fix (non-NULL garbage value Pointer)]** I then ran into another segmentation fault and found that the for loop at the end of the create_LL function was iterating one too many times. To fix this I had it iterate while 'i < node_num - 1'.



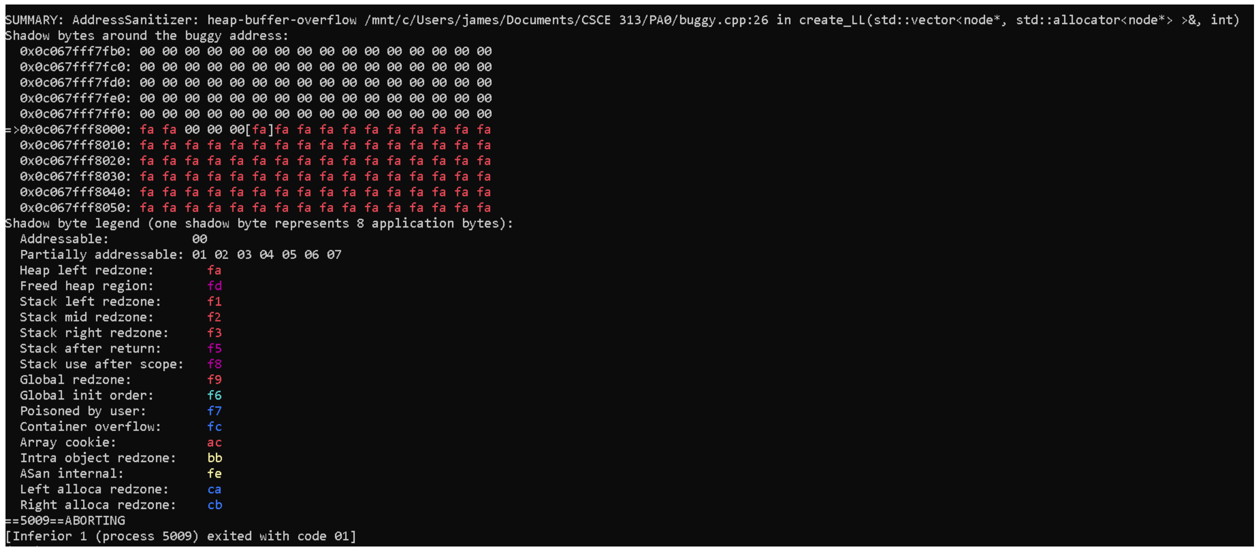
```
Program received signal SIGSEGV, Segmentation fault.
0x0000000008000cf6 in sum_LL (ptr=0x21) at buggy.cpp:33
33                  ret += ptr->val;
```

8. **[Deletion of Dynamically Allocated Memory]** I then wrote code using the 'delete' keyword to free up the memory that was allocated.



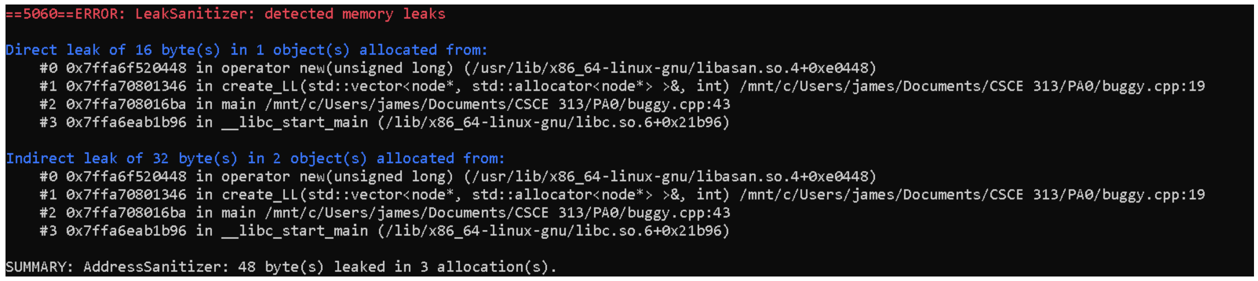
```
//Step4: delete nodes
//Blank D
for (int i = 0; i < NODE_NUM; i++) {
    delete mylist[i];
}
```

9. **[AddressSanitizer]** I then repeated these steps while using AddressSanitizer and got different error messages that helped me understand how memory leaks were occurring. I fixed the same compilation errors and then got run time errors from AddressSanitizer.

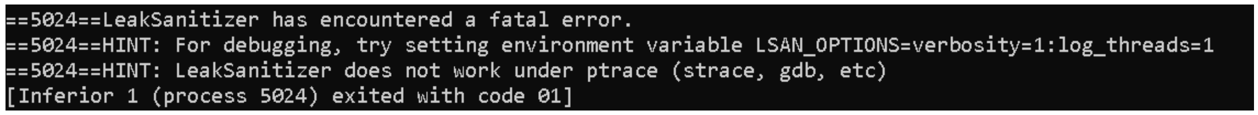


```
SUMMARY: AddressSanitizer: heap-buffer-overflow /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy.cpp:26 in create_LL(std::vector<node*, std::allocator<node*> >&, int)
Shadow bytes around the buggy address:
  0x0c067fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c067fff8000: fa fa 00 00 00[fa]fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==5009==ABORTING
[Inferior 1 (process 5009) exited with code 01]
```

I instantiated the pointers to the nodes in the linked list but got the below output when I did not delete them.

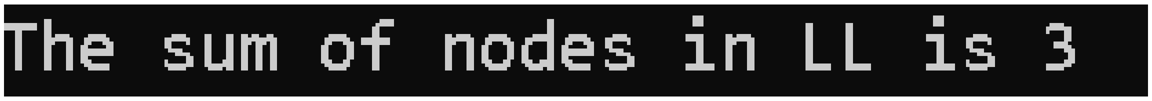


```
==5060==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7ffa6f520448 in operator new(unsigned long) (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xe0448)
    #1 0x7ffa70801346 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy.cpp:19
    #2 0x7ffa708016ba in main /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy.cpp:43
    #3 0x7ffa6eab1b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

Indirect leak of 32 byte(s) in 2 object(s) allocated from:
    #0 0x7ffa6f520448 in operator new(unsigned long) (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xe0448)
    #1 0x7ffa70801346 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy.cpp:19
    #2 0x7ffa708016ba in main /mnt/c/Users/james/Documents/CSCE 313/PA0/buggy.cpp:43
    #3 0x7ffa6eab1b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: 48 byte(s) leaked in 3 allocation(s).
```



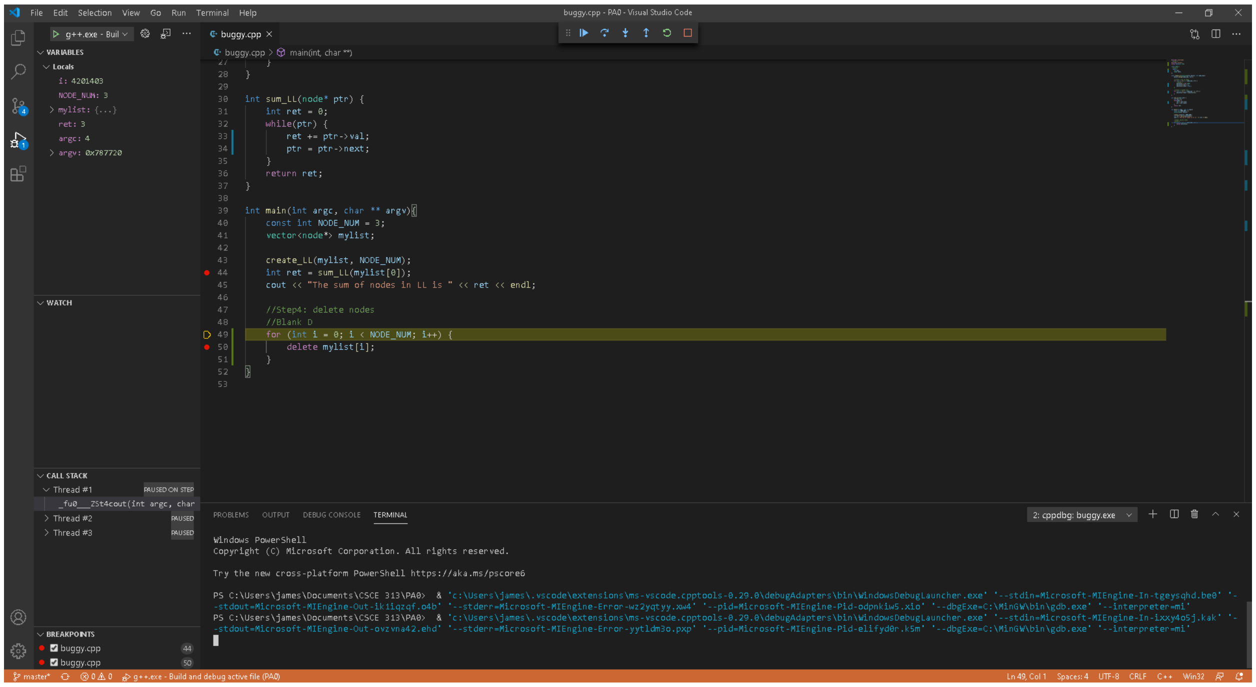
```
==5024==LeakSanitizer has encountered a fatal error.
==5024==HINT: For debugging, try setting environment variable LSAN_OPTIONS=verbosity=1:log_threads=1
==5024==HINT: LeakSanitizer does not work under ptrace (strace, gdb, etc)
[Inferior 1 (process 5024) exited with code 01]
```

But with the completed correct code, gdb with and without AddressSanitizer outputted the same thing.



```
The sum of nodes in LL is 3
```

10. [Using IDE] Here is a picture of VSCode while I was debugging the program.



File Edit Selection View Go Run Terminal Help
buggy.cpp - PA0 - Visual Studio Code
g++.exe - Buil
buggy.cpp
buggy.cpp > main(int, char **)
VARIABLES
Locals
WATCH
CALL STACK
Thread #1 PAUSED ON STEP
Thread #2 PAUSED
Thread #3 PAUSED
BREAKPOINTS
buggy.cpp 44
buggy.cpp 50
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: cppdbg: buggy.exe
master* 0 0 g++.exe - Build and debug active file (PA0)
Ln 49, Col 1 Spaces: 4 UTF-8 CRLF C++ Win32

```
i: 4201403
NODE_NUM: 3
> mylist: {...}
ret: 3
argc: 4
> argv: 0x787720
```

```
_fu0___ZSt4cout(int argc, char
```

```cpp
    }
}

int sum_LL(node* ptr) {
    int ret = 0;
    while(ptr) {
        ret += ptr->val;
        ptr = ptr->next;
    }
    return ret;
}

int main(int argc, char ** argv){
    const int NODE_NUM = 3;
    vector<node*> mylist;

    create_LL(mylist, NODE_NUM);
    int ret = sum_LL(mylist[0]);
    cout << "The sum of nodes in LL is " << ret << endl;

    //Step4: delete nodes
    //Blank D
    for (int i = 0; i < NODE_NUM; i++) {
        delete mylist[i];
    }
}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\james\Documents\CSCE 313\PA0>  & 'c:\Users\james\.vscode\extensions\ms-vscode.cpptools-0.29.0\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-tgeysqhd.be0' '--stdout=Microsoft-MIEngine-Out-ik1iqzqf.o4b' '--stderr=Microsoft-MIEngine-Error-wz2yqtyy.xw4' '--pid=Microsoft-MIEngine-Pid-odpnkiw5.xio' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
PS C:\Users\james\Documents\CSCE 313\PA0>  & 'c:\Users\james\.vscode\extensions\ms-vscode.cpptools-0.29.0\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ixxy4o5j.kak' '--stdout=Microsoft-MIEngine-Out-ovzvna42.ehd' '--stderr=Microsoft-MIEngine-Error-yytldm3o.pxp' '--pid=Microsoft-MIEngine-Pid-elifyd0r.k5m' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'
```