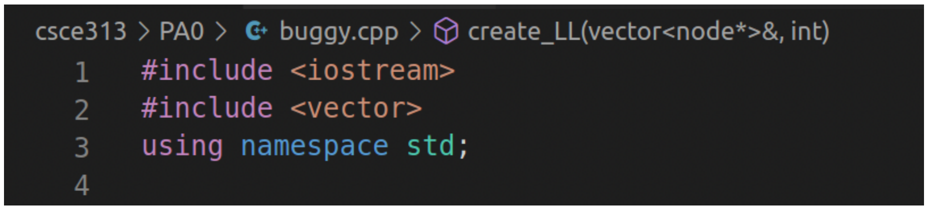
Raul Escobar
Csce 313-599
328003859
9/3/2021

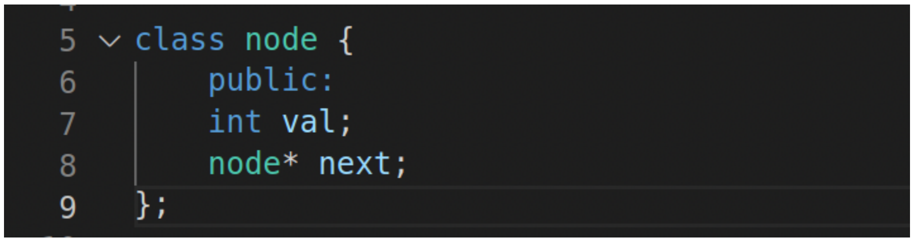
Programming Assignment 0 Report

1. Compile Time errors
   a. First change:
      i. #include <vector>
      ii. using namespace as std;
      iii. because the compiler could not recognize the vector instantiation and several lines in the script were using cout and endl without the std::



```
csce313 > PA0 > buggy.cpp > create_LL(vector<node*>&, int)
#include <iostream>
#include <vector>
using namespace std;

```

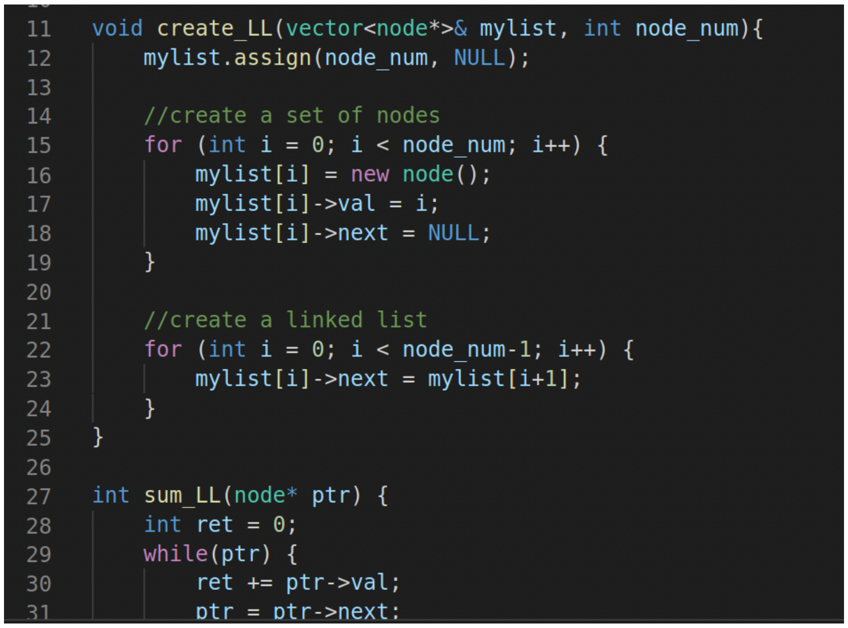
   b. Second change:
      i. Set node member variables to public so that the main function could access them



```
class node {
    public:
    int val;
    node* next;
};
```

   c. Third change:
      i. Replace . operator with -> because pointer was not being dereferenced when accessed.

Raul Escobar
Csce 313-599
328003859
9/3/2021

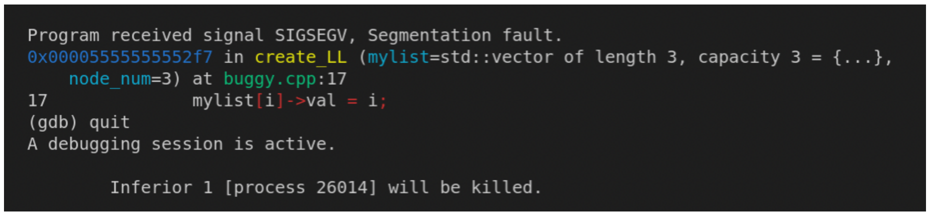


```
11  void create_LL(vector<node*>& mylist, int node_num){
12      mylist.assign(node_num, NULL);
13  
14      //create a set of nodes
15      for (int i = 0; i < node_num; i++) {
16          mylist[i] = new node();
17          mylist[i]->val = i;
18          mylist[i]->next = NULL;
19      }
20  
21      //create a linked list
22      for (int i = 0; i < node_num-1; i++) {
23          mylist[i]->next = mylist[i+1];
24      }
25  }
26  
27  int sum_LL(node* ptr) {
28      int ret = 0;
29      while(ptr) {
30          ret += ptr->val;
31          ptr = ptr->next;
```

d. After these changes the script would compile, but when the executable was run it would return a segmentation fault so we have to use gdb/address sanitizer/ide debugger to find the problem

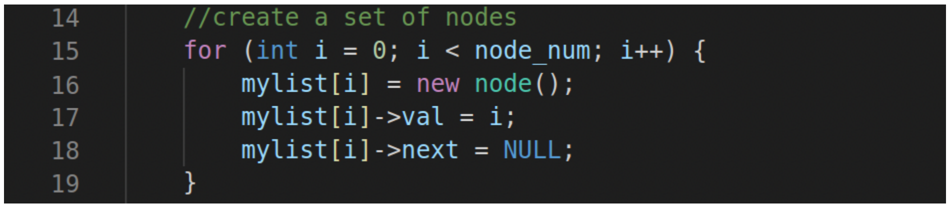
2. Runtime errors

- After running gdb i get the error:



```
Program received signal SIGSEGV, Segmentation fault.
0x00005555555552f7 in create_LL (mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:17
17              mylist[i]->val = i;
(gdb) quit
A debugging session is active.

        Inferior 1 [process 26014] will be killed.
```

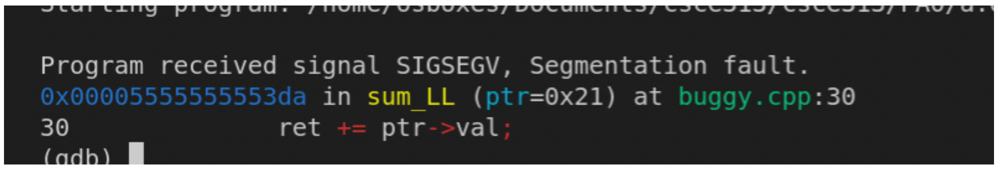
- Added the line mylist[i] = new node(); when creating the linked list because line 17 was trying to access a node that had not been instantiated yet. I knew this because sigsegv signals are for “trying to read or write from/to a memory area that your process does not have access to”



```
14      //create a set of nodes
15      for (int i = 0; i < node_num; i++) {
16          mylist[i] = new node();
17          mylist[i]->val = i;
18          mylist[i]->next = NULL;
19      }
```

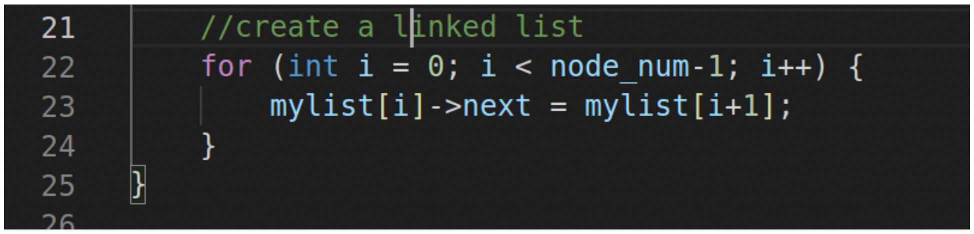
- Then after running again i get:

Raul Escobar
Csce 313-599
328003859
9/3/2021

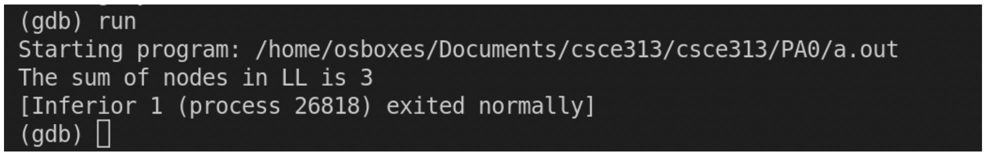


```
Starting program: /home/osboxes/Documents/csce313/csce313/PA0/a.

Program received signal SIGSEGV, Segmentation fault.
0x00005555555553da in sum_LL (ptr=0x21) at buggy.cpp:30
30              ret += ptr->val;
(gdb) 
```

- This is because the last node of linked list is pointing to node that does not exist so we fix the for loop arguments so that it stops before the last node and recompile to get

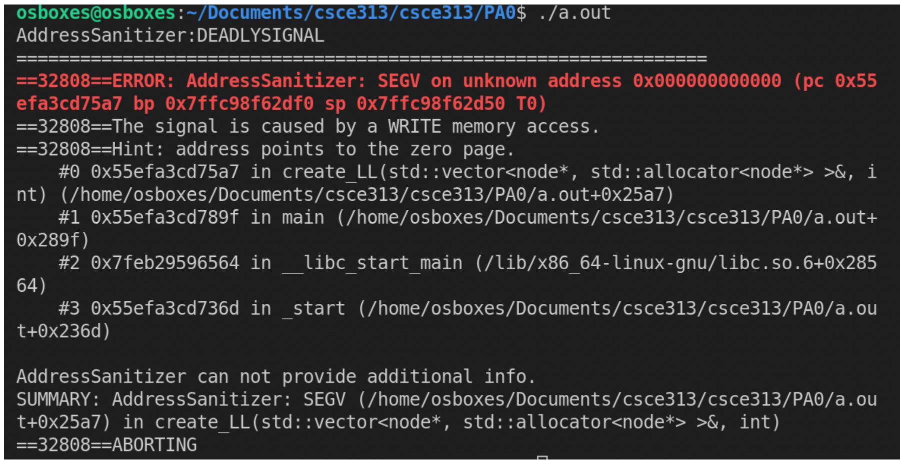


```
    //create a linked list
    for (int i = 0; i < node_num-1; i++) {
        mylist[i]->next = mylist[i+1];
    }
}
```



```
(gdb) run
Starting program: /home/osboxes/Documents/csce313/csce313/PA0/a.out
The sum of nodes in LL is 3
[Inferior 1 (process 26818) exited normally]
(gdb) 
```

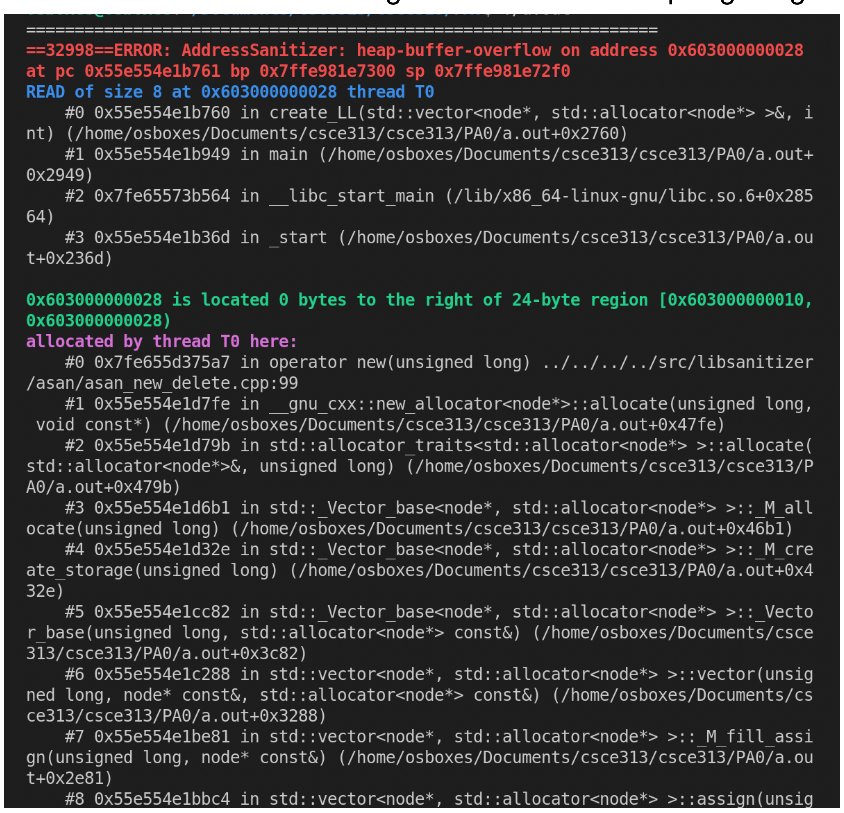
- I reset the fixes I made while using gdb to solve the issues with fsanitize, and when I ran the script I got the output:



```
osboxes@osboxes:~/Documents/csce313/csce313/PA0$ ./a.out
AddressSanitizer:DEADLYSIGNAL
=================================================================
==32808==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x55
efa3cd75a7 bp 0x7ffc98f62df0 sp 0x7ffc98f62d50 T0)
==32808==The signal is caused by a WRITE memory access.
==32808==Hint: address points to the zero page.
    #0 0x55efa3cd75a7 in create_LL(std::vector<node*, std::allocator<node*> >&, i
nt) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x25a7)
    #1 0x55efa3cd789f in main (/home/osboxes/Documents/csce313/csce313/PA0/a.out+
0x289f)
    #2 0x7feb29596564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x285
64)
    #3 0x55efa3cd736d in _start (/home/osboxes/Documents/csce313/csce313/PA0/a.ou
t+0x236d)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV (/home/osboxes/Documents/csce313/csce313/PA0/a.ou
t+0x25a7) in create_LL(std::vector<node*, std::allocator<node*> >&, int)
==32808==ABORTING
```

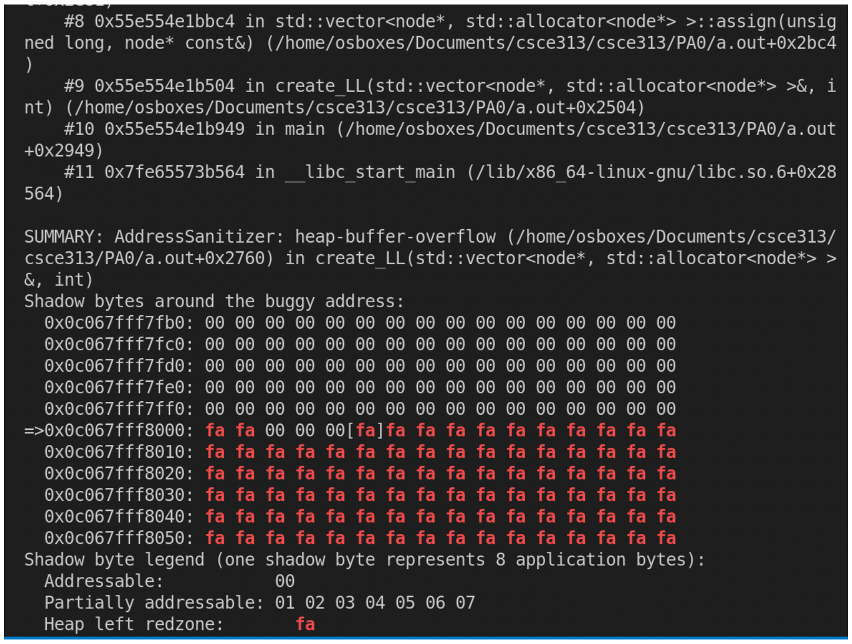
Raul Escobar
Csce 313-599
328003859
9/3/2021

- We already know what the error is and how to fix it because of gdb, but we can see how fsanitize also tells us the same thing, but in a different way
- We can see that the hints tell us the error comes from write memory access, and that address points to the zero page.
- Then we can see that it originates from the function create_LL
- We only write to memory in create_LL in 3 lines so after further investigation it is not too crazy to think the writer of the program can spot that the node being written to has not been created yet.
- Then after fixing the error and recompiling we get the error:



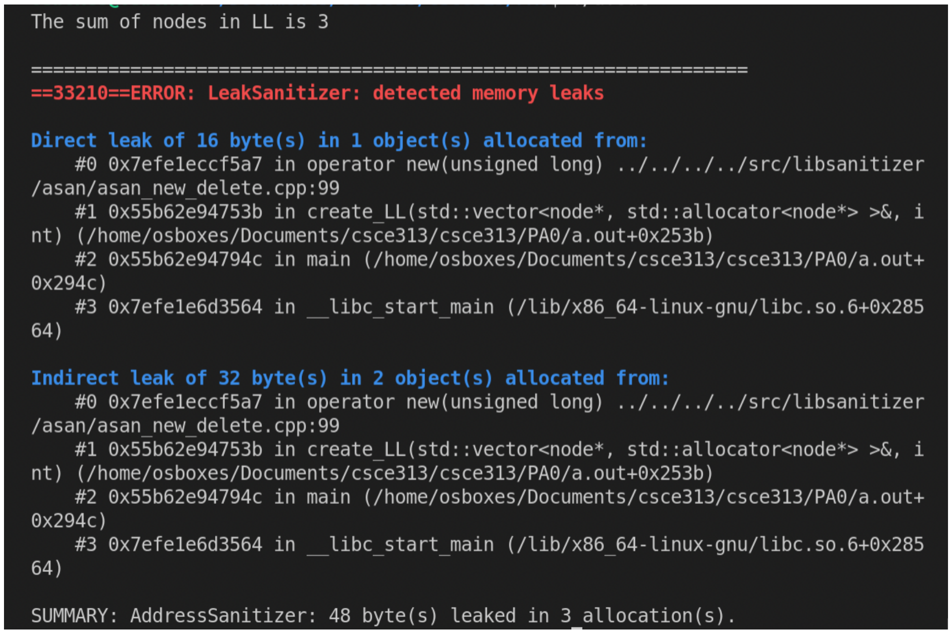
```
=================================================================
==32998==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000000028
at pc 0x55e554e1b761 bp 0x7ffe981e7300 sp 0x7ffe981e72f0
READ of size 8 at 0x603000000028 thread T0
    #0 0x55e554e1b760 in create_LL(std::vector<node*, std::allocator<node*> >&, i
nt) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x2760)
    #1 0x55e554e1b949 in main (/home/osboxes/Documents/csce313/csce313/PA0/a.out+
0x2949)
    #2 0x7fe65573b564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x285
64)
    #3 0x55e554e1b36d in _start (/home/osboxes/Documents/csce313/csce313/PA0/a.ou
t+0x236d)

0x603000000028 is located 0 bytes to the right of 24-byte region [0x603000000010,
0x603000000028)
allocated by thread T0 here:
    #0 0x7fe655d375a7 in operator new(unsigned long) ../../../../src/libsanitizer
/asan/asan_new_delete.cpp:99
    #1 0x55e554e1d7fe in __gnu_cxx::new_allocator<node*>::allocate(unsigned long,
 void const*) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x47fe)
    #2 0x55e554e1d79b in std::allocator_traits<std::allocator<node*> >::allocate(
std::allocator<node*>&, unsigned long) (/home/osboxes/Documents/csce313/csce313/P
A0/a.out+0x479b)
    #3 0x55e554e1d6b1 in std::_Vector_base<node*, std::allocator<node*> >::_M_all
ocate(unsigned long) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x46b1)
    #4 0x55e554e1d32e in std::_Vector_base<node*, std::allocator<node*> >::_M_cre
ate_storage(unsigned long) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x4
32e)
    #5 0x55e554e1cc82 in std::_Vector_base<node*, std::allocator<node*> >::_Vecto
r_base(unsigned long, std::allocator<node*> const&) (/home/osboxes/Documents/csce
313/csce313/PA0/a.out+0x3c82)
    #6 0x55e554e1c288 in std::vector<node*, std::allocator<node*> >::vector(unsig
ned long, node* const&, std::allocator<node*> const&) (/home/osboxes/Documents/cs
ce313/csce313/PA0/a.out+0x3288)
    #7 0x55e554e1be81 in std::vector<node*, std::allocator<node*> >::_M_fill_assi
gn(unsigned long, node* const&) (/home/osboxes/Documents/csce313/csce313/PA0/a.ou
t+0x2e81)
    #8 0x55e554e1bbc4 in std::vector<node*, std::allocator<node*> >::assign(unsig
```

Raul Escobar
Csce 313-599
328003859
9/3/2021



```
    #8 0x55e554e1bbc4 in std::vector<node*, std::allocator<node*> >::assign(unsig
ned long, node* const&) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x2bc4
)
    #9 0x55e554e1b504 in create_LL(std::vector<node*, std::allocator<node*> >&, i
nt) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x2504)
    #10 0x55e554e1b949 in main (/home/osboxes/Documents/csce313/csce313/PA0/a.out
+0x2949)
    #11 0x7fe65573b564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x28
564)

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/osboxes/Documents/csce313/
csce313/PA0/a.out+0x2760) 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
```

- Again the error originates from create_LL and seems to be an out of bounds access.
- This is because in the for loop where we point the node to the next node, we include the last node, which is supposed to be pointing to NULL.
- We can fix this by making sure the for loop does not reach the last node.

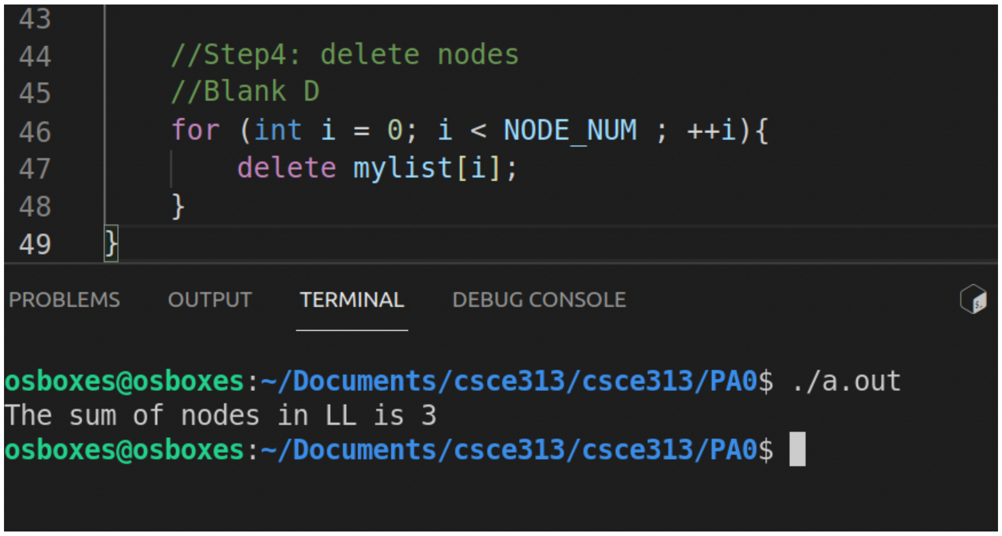


```
The sum of nodes in LL is 3

=================================================================
==33210==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7efe1eccf5a7 in operator new(unsigned long) ../../../../src/libsanitizer
/asan/asan_new_delete.cpp:99
    #1 0x55b62e94753b in create_LL(std::vector<node*, std::allocator<node*> >&, i
nt) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x253b)
    #2 0x55b62e94794c in main (/home/osboxes/Documents/csce313/csce313/PA0/a.out+
0x294c)
    #3 0x7efe1e6d3564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x285
64)

Indirect leak of 32 byte(s) in 2 object(s) allocated from:
    #0 0x7efe1eccf5a7 in operator new(unsigned long) ../../../../src/libsanitizer
/asan/asan_new_delete.cpp:99
    #1 0x55b62e94753b in create_LL(std::vector<node*, std::allocator<node*> >&, i
nt) (/home/osboxes/Documents/csce313/csce313/PA0/a.out+0x253b)
    #2 0x55b62e94794c in main (/home/osboxes/Documents/csce313/csce313/PA0/a.out+
0x294c)
    #3 0x7efe1e6d3564 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x285
64)

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

- This error comes from memory leaks according to f-sanitize.
- There are 48 bytes leakes from 3 allocations.
- This makes sense because we are allocating 3 nodes, but we never delete them

Raul Escobar
Csce 313-599
328003859
9/3/2021

- We just have to delete the nodes we created in the main function and recompile.

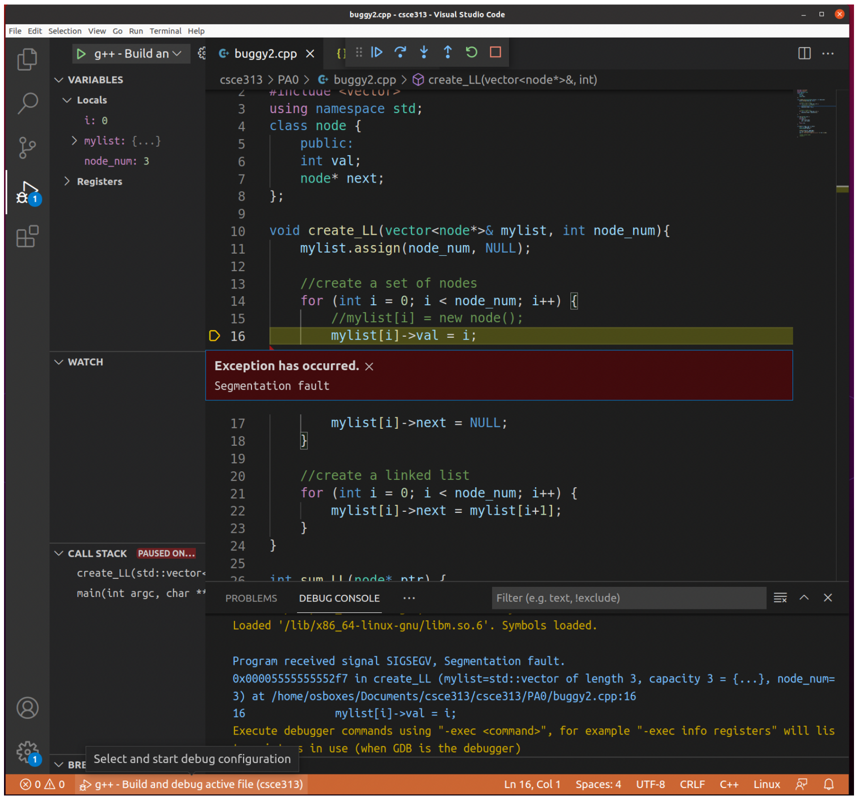


```

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

```
osboxes@osboxes:~/Documents/csce313/csce313/PA0$ ./a.out
The sum of nodes in LL is 3
osboxes@osboxes:~/Documents/csce313/csce313/PA0$
```

- When running the IDE debugger for visual studios code i get the following messages after fixing the compile time errors.

Raul Escobar
Csce 313-599
328003859
9/3/2021



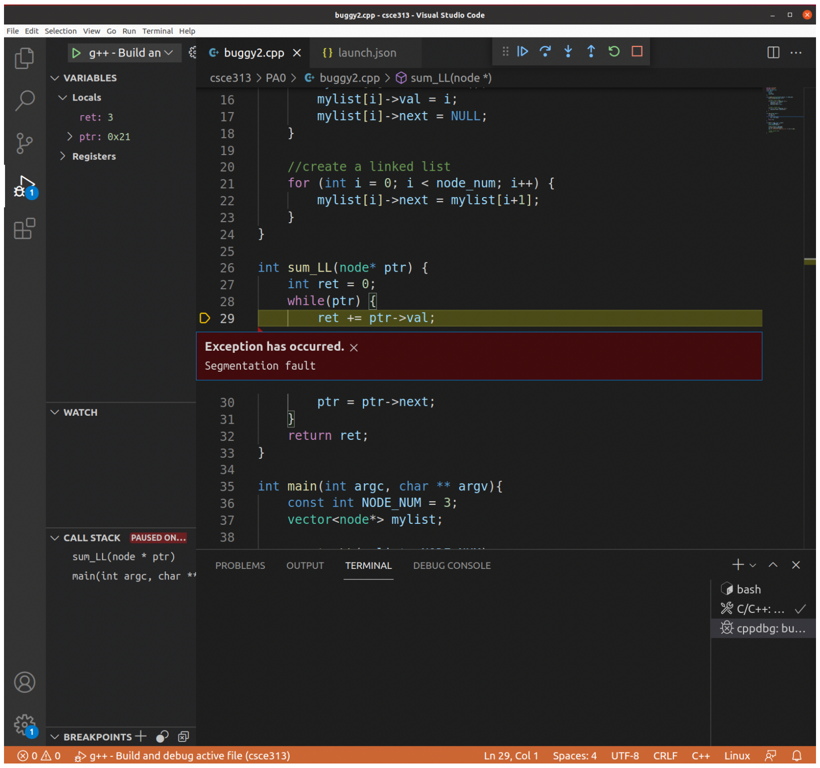
```
#include <vector>
using namespace std;
class node {
    public:
    int val;
    node* next;
};

void create_LL(vector<node*>& mylist, int node_num){
    mylist.assign(node_num, NULL);

    //create a set of nodes
    for (int i = 0; i < node_num; i++) {
        //mylist[i] = new node();
        mylist[i]->val = i;
```

Exception has occurred.
Segmentation fault

```
        mylist[i]->next = NULL;
    }

    //create a linked list
    for (int i = 0; i < node_num; i++) {
        mylist[i]->next = mylist[i+1];
    }
}

int sum_LL(node* ptr) {
```

```
Loaded '/lib/x86_64-linux-gnu/libm.so.6'. Symbols loaded.

Program received signal SIGSEGV, Segmentation fault.
0x00005555555552f7 in create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=
3) at /home/osboxes/Documents/csce313/csce313/PA0/buggy2.cpp:16
16              mylist[i]->val = i;
Execute debugger commands using "-exec <command>", for example "-exec info registers" will lis
          s in use (when GDB is the debugger)
```

buggy2.cpp - csce313 - Visual Studio Code
g++ - Build an
buggy2.cpp
csce313 > PA0 > buggy2.cpp > create_LL(vector<node*>&, int)
VARIABLES
Locals
i: 0
mylist: {...}
node_num: 3
Registers
WATCH
CALL STACK PAUSED ON...
create_LL(std::vector
main(int argc, char **
PROBLEMS
DEBUG CONSOLE
Select and start debug configuration
g++ - Build and debug active file (csce313)
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF C++ Linux



```
        mylist[i]->val = i;
        mylist[i]->next = NULL;
    }

    //create a linked list
    for (int i = 0; i < node_num; i++) {
        mylist[i]->next = mylist[i+1];
    }
}

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

Exception has occurred.
Segmentation fault

```
        ptr = ptr->next;
    }
    return ret;
}

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

buggy2.cpp - csce313 - Visual Studio Code
g++ - Build an
buggy2.cpp
launch.json
csce313 > PA0 > buggy2.cpp > sum_LL(node *)
VARIABLES
Locals
ret: 3
ptr: 0x21
Registers
WATCH
CALL STACK PAUSED ON...
sum_LL(node * ptr)
main(int argc, char **
BREAKPOINTS
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
C/C++: ...
cppdbg: bu...
g++ - Build and debug active file (csce313)
Ln 29, Col 1 Spaces: 4 UTF-8 CRLF C++ Linux

- This gives us the same errors as before with the same fixes.

Raul Escobar
Csce 313-599
328003859
9/3/2021

After fixing this, it did not give any more errors after rerunning.