Arturo Villalobos

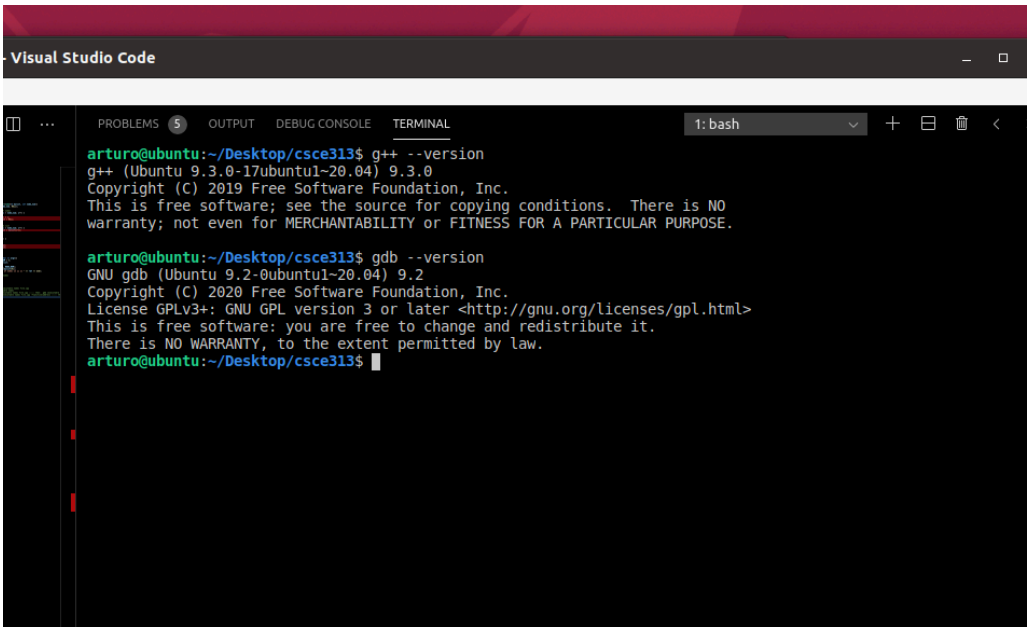
CSCE 313-503

Professor Sarker Tanzir Ahmed

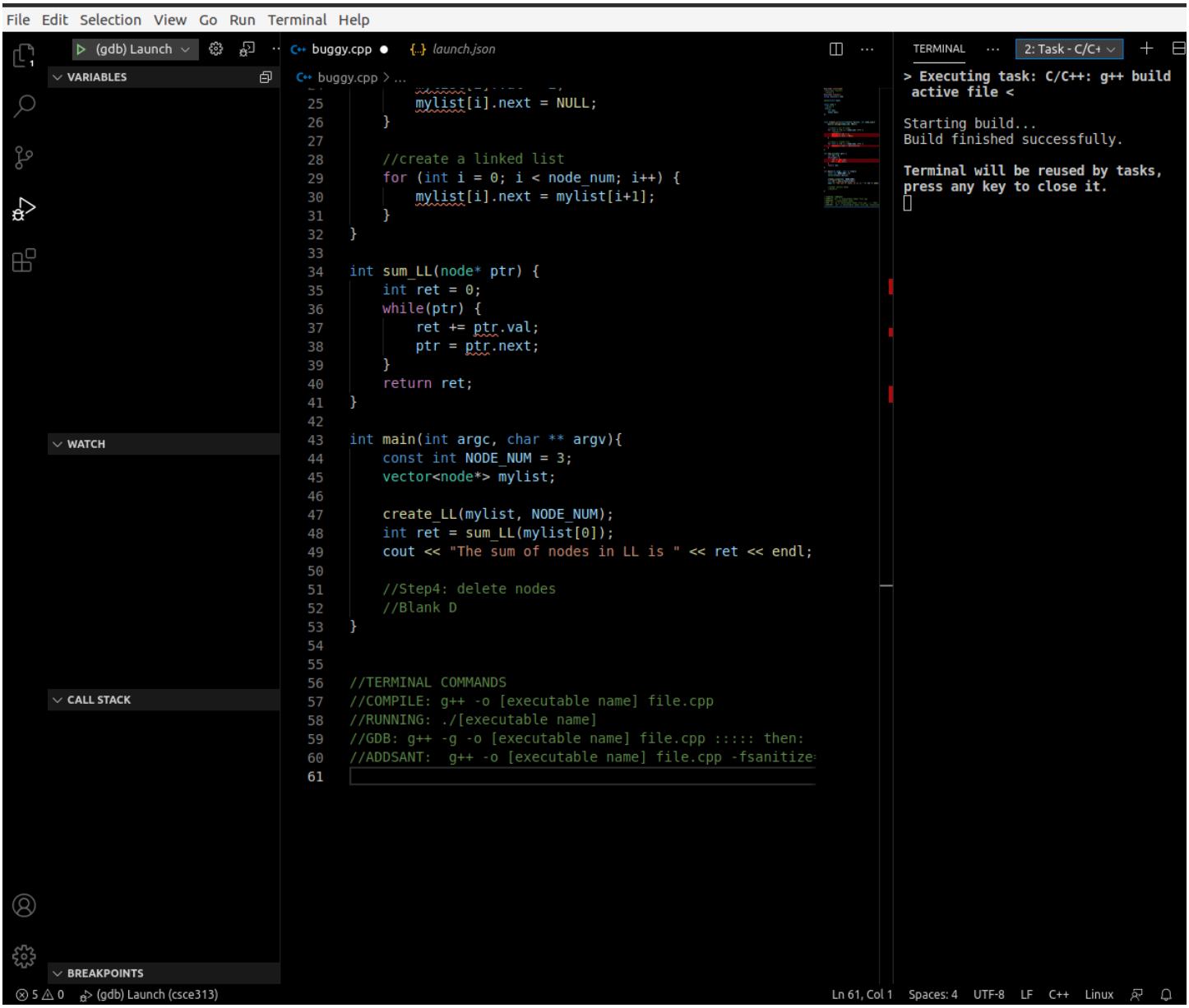
Jan 29 2021

*Programming Assignment : 0*

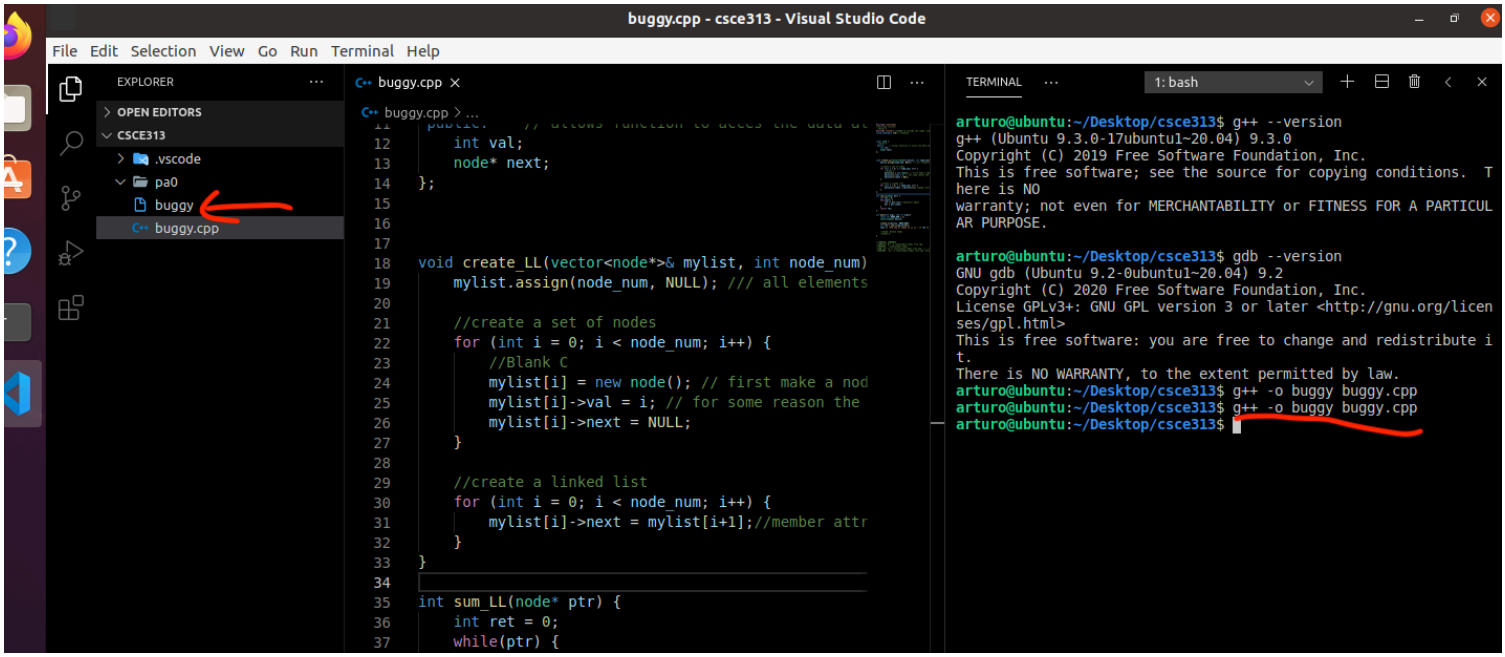
1 &10.) For system setup , I downloaded the latest version of ubuntu and loaded it into a virtual machine using Vmware fusion. The screens shot below show that I currently have the last version of g++ along with GDB and Address Sanitizer.




I also installed the most rated debugger for c++ shown below




2. After correcting the issues of including the vector library and making the node class have public attributes along with with lines 15, 16, 21, 28, 29, I was able to compile the cpp file into an executable called buggy as shown below




3 & 4. To use gdb I used the same compile command as the one shown above , but now I just added the -g flag the the terminal command. From marked screen shot below you can see that I combined two steps by first running the program, setting a break point at (in my case 24) to see what happens when we try to access a list full of null pointers. After using the ’n’ command I run into a segmentation fault on line 25 where then I print the entire list to see why that would be. Using the back trace I can see that the the error is triggered when I call the create_LL function , and more specifically on line 25 within the function itself. This is because every element is set to null, so when try to access data attributes there is nothing there. To fix this I just added a ‘mylist[i] = new node();’ statement above this to actually have nodes in the list.

```
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp -g
arturo@ubuntu:~/Desktop/csce313$ gdb buggy
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licen
ses/gpl.html>
This is free software: you are free to change and redistribute i
t.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from buggy...
(gdb) run
Starting program: /home/arturo/Desktop/csce313/buggy

Program received signal SIGSEGV, Segmentation fault.
0x00005555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:25
25              mylist[i]->val = i; // for some reason the . ope
rator didnt work so i need to use ->
(gdb) b 24
Breakpoint 1 at 0x5555555552dc: file buggy.cpp, line 25.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/arturo/Desktop/csce313/buggy

Breakpoint 1, create_LL (
    mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:25
25              mylist[i]->val = i; // for some reason the . ope
rator didnt work so i need to use ->
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x00005555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:25
25              mylist[i]->val = i; // for some reason the . ope
rator didnt work so i need to use ->
(gdb) print mylist
$1 = std::vector of length 3, capacity 3 = {0x0, 0x0, 0x0}
(gdb) stackframe
Undefined command: "stackframe".  Try "help".
(gdb) backtrace
#0  0x00005555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:25
#1  0x0000555555555407 in main (argc=1, argv=0x7fffffffdf98)
    at buggy.cpp:48
(gdb)
```

Ln 24, Col 11 Spaces: 4 UTF-8 LF C++ Linux

5,6, & 7. After inserting this statement and running the program , we get this output:

```
(gdb) n
30          for (int i = 0; i < node_num; i++) {
(gdb) print mylist
$2 = std::vector of length 3, capacity 3 = {0x55555556bed0,
  0x55555556bef0, 0x55555556bf10}
(gdb)
```

Ln 24, Col 9  Spaces: 4  UTF-8  LF  C++  Linux

This is different from what we had before where all the items in the list where null. This is all good but if we run the program in gdb fresh I will get a seffualt on line 4324 in sum_LL . The hint points out that the second part of the create function is wrong. After setting a break point on line 29 , I realized that mylist[i]->next = mylist[I+1] would run even though the loop was on the last node , meaning the last node's next pointer would be referencing some unallocated part of memory. After adding this check shown below , I was able to run the program with not segmentation faults.

buggy.cpp > create_LL(vector<node*>&, int)

```
17
18  void create_LL(vector<node*>& mylist, int node
19      mylist.assign(node_num, NULL); /// all ele
20
21      //create a set of nodes
22      for (int i = 0; i < node_num; i++) {
23          //Blank C
24          mylist[i] = new node(); // first make
25          mylist[i]->val = i; // for some reason
26          mylist[i]->next = NULL;
27      }
28
29      //create a linked list
30      for (int i = 0; i < node_num; i++) {
31          if(i == node_num-1)
32          {
33              mylist[i]->next = NULL;
34          }
35          else
36          {
37              mylist[i]->next = mylist[i+1];//me
38          }
39
40
41      }
42  }
43
44  int sum_LL(node* ptr) {
45      int ret = 0;
```

```
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy b
uggy.cpp -g
arturo@ubuntu:~/Desktop/csce313$ gdb buggy
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, In
c.
License GPLv3+: GNU GPL version 3 or later <htt
p://gnu.org/licenses/gpl.html>
This is free software: you are free to change a
nd redistribute it.
There is NO WARRANTY, to the extent permitted b
y law.
Type "show copying" and "show warranty" for det
ails.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration det
ails.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation res
ources online at:
    <http://www.gnu.org/software/gdb/documentat
ion/>.

For help, type "help".
Type "apropos word" to search for commands rela
ted to "word"...
Reading symbols from buggy...
(gdb) run
Starting program: /home/arturo/Desktop/csce313/
buggy
The sum of nodes in LL is 3
[Inferior 1 (process 12673) exited normally]
(gdb)
```

8 & 9. Running the program using address sanitizer and making no edits I got this output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   1: bash

arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp -fsanitize=address
arturo@ubuntu:~/Desktop/csce313$ ./buggy
The sum of nodes in LL is 3
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp -fsanitize=address
arturo@ubuntu:~/Desktop/csce313$ ./buggy
The sum of nodes in LL is 3

=================================================================
==12920==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7fad69caa947 in operator new(unsigned long) (/lib/x86_64-linux-gnu/libasan.s
o.5+0x10f947)
    #1 0x55e85f84e53d in create_LL(std::vector<node*, std::allocator<node*> >&, int)
(/home/arturo/Desktop/csce313/buggy+0x253d)
    #2 0x55e85f84e9c8 in main (/home/arturo/Desktop/csce313/buggy+0x29c8)
    #3 0x7fad697d40b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

Indirect leak of 32 byte(s) in 2 object(s) allocated from:
    #0 0x7fad69caa947 in operator new(unsigned long) (/lib/x86_64-linux-gnu/libasan.s
o.5+0x10f947)
    #1 0x55e85f84e53d in create_LL(std::vector<node*, std::allocator<node*> >&, int)
(/home/arturo/Desktop/csce313/buggy+0x253d)
    #2 0x55e85f84e9c8 in main (/home/arturo/Desktop/csce313/buggy+0x29c8)
    #3 0x7fad697d40b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

SUMMARY: AddressSanitizer: 48 byte(s) leaked in 3 allocation(s).
arturo@ubuntu:~/Desktop/csce313$
```

The output points out that I have 3 leaked allocations. These memory leaks correspond to the nodes on the list I have not felt with yet. To deal with this I made a loop to iterate through the vector and delete each node. After making this fix and running address sanitizer again they're where no leaked allocations remaining . The fix and output are shown below:

```
    return ret;
}

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

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

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

std::vector<node *> mylist

```
SUMMARY: AddressSanitizer: 48 byte(s) leaked in 3 allocation(s).
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp -fsanitize=address
arturo@ubuntu:~/Desktop/csce313$ ./buggy
The sum of nodes in LL is 3
arturo@ubuntu:~/Desktop/csce313$
```