

Arturo Villalobos

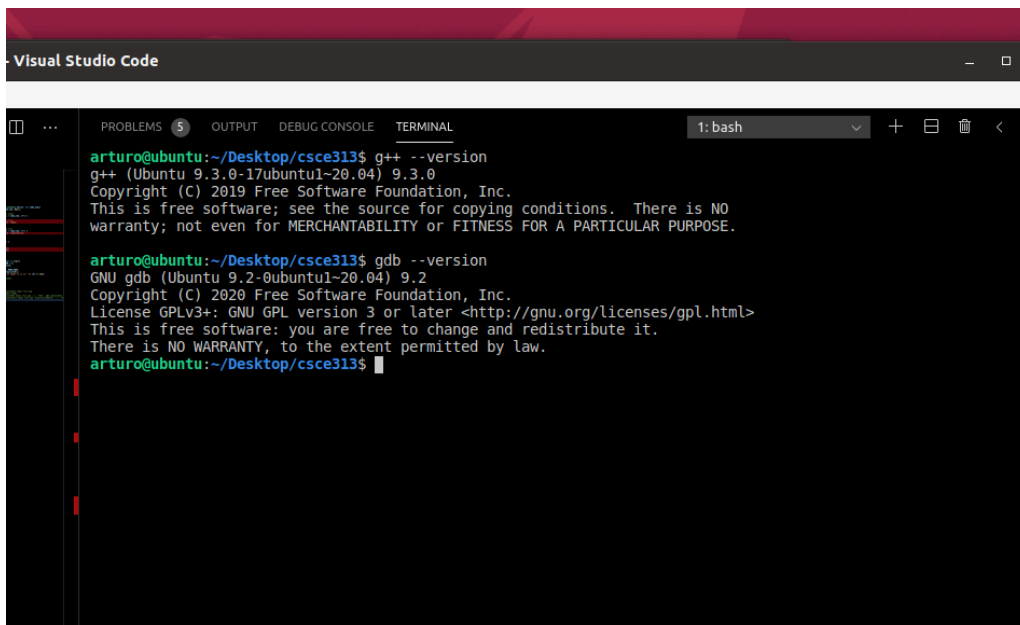
CSCE 313-503

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.

A screenshot of the Visual Studio Code interface. The top bar shows 'Visual Studio Code' and window controls. The main area is a terminal window titled '1: bash'. The terminal shows the following commands and output:

```
arturo@ubuntu:~/Desktop/csce313$ g++ --version
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

arturo@ubuntu:~/Desktop/csce313$ gdb --version
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/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
arturo@ubuntu:~/Desktop/csce313$
```

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

The screenshot shows a C++ IDE with a dark theme. The main editor displays the source code for `buggy.cpp`. The code defines a linked list structure and a function to calculate the sum of its nodes. The terminal on the right shows the successful compilation and execution of the program.

```
File Edit Selection View Go Run Terminal Help

C++ buggy.cpp • launch.json

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

buggy.cpp > ...
25     mylist[i].next = NULL;
26 }
27
28 //create a linked list
29 for (int i = 0; i < node_num; i++) {
30     mylist[i].next = mylist[i+1];
31 }
32 }
33
34 int sum_LL(node* ptr) {
35     int ret = 0;
36     while(ptr) {
37         ret += ptr.val;
38         ptr = ptr.next;
39     }
40     return ret;
41 }
42
43 int main(int argc, char ** argv){
44     const int NODE_NUM = 3;
45     vector<node*> mylist;
46
47     create_LL(mylist, NODE_NUM);
48     int ret = sum_LL(mylist[0]);
49     cout << "The sum of nodes in LL is " << ret << endl;
50
51     //Step4: delete nodes
52     //Blank D
53 }
54
55
56 //TERMINAL COMMANDS
57 //COMPILE: g++ -o [executable name] file.cpp
58 //RUNNING: ./[executable name]
59 //GDB: g++ -g -o [executable name] file.cpp :::: then:
60 //ADDSANT: g++ -o [executable name] file.cpp -fsanitize:
61
```

TERMINAL 2: Task - C/C++

```
> Executing task: C/C++: g++ build
active file <

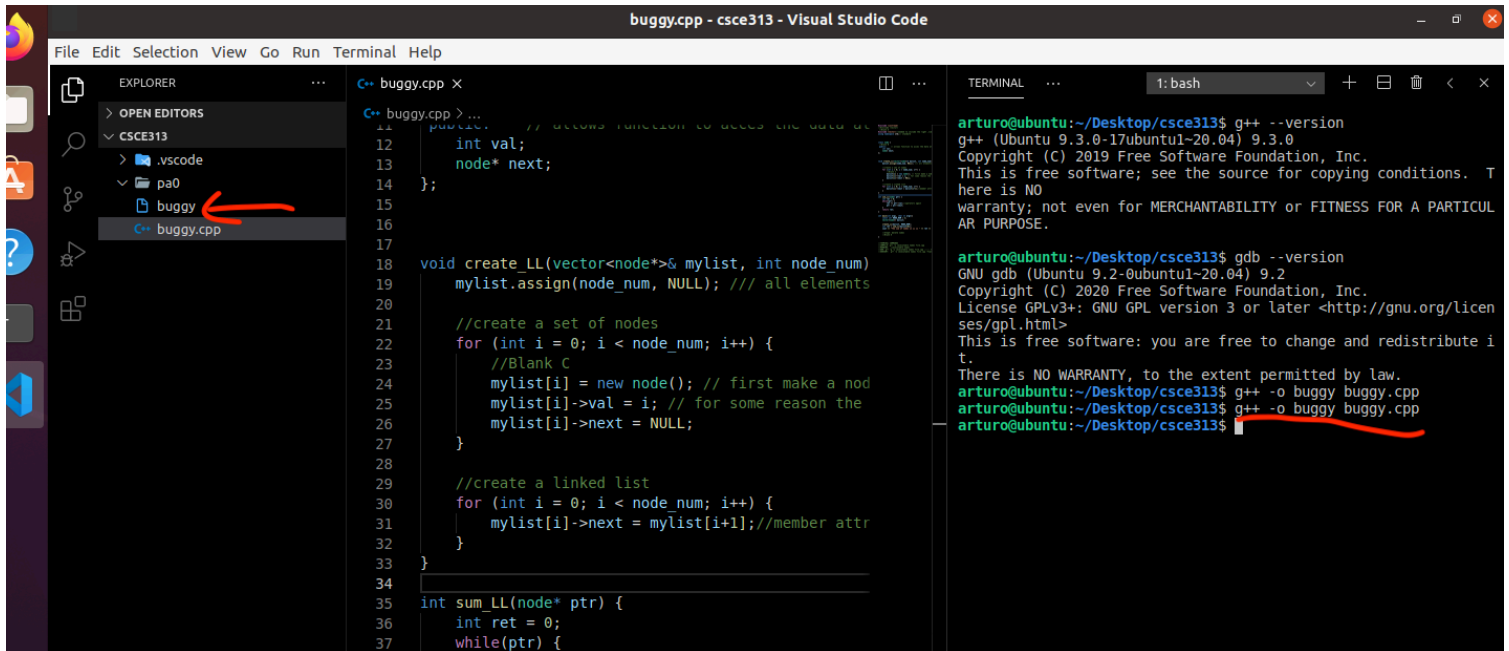
Starting build...
Build finished successfully.

Terminal will be reused by tasks,
press any key to close it.

```

Ln 61, Col 1 Spaces: 4 UTF-8 LF C++ Linux

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



```
File Edit Selection View Go Run Terminal Help
EXPLORER
> OPEN EDITORS
CSCE313
  .vscode
  pa0
  buggy
  C++ buggy.cpp

C++ buggy.cpp x
11 public: // allows function to access the data of
12     int val;
13     node* next;
14 };
15
16
17
18 void create_LL(vector<node*>& mylist, int node_num)
19     mylist.assign(node_num, NULL); /// all elements
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 a nod
25         mylist[i]->val = i; // for some reason the
26         mylist[i]->next = NULL;
27     }
28
29     //create a linked list
30     for (int i = 0; i < node_num; i++) {
31         mylist[i]->next = mylist[i+1]; //member attr
32     }
33 }
34
35 int sum_LL(node* ptr) {
36     int ret = 0;
37     while(ptr) {
```

```
arturo@ubuntu:~/Desktop/csce313$ g++ --version
g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. T
here is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICU
AR PURPOSE.

arturo@ubuntu:~/Desktop/csce313$ gdb --version
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.
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp
arturo@ubuntu:~/Desktop/csce313$ g++ -o buggy buggy.cpp
arturo@ubuntu:~/Desktop/csce313$
```

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/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
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.
0x0000555555552f7 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 . operator
    rator didnt work so i need to use ->
(gdb) b 24
Breakpoint 1 at 0x555555552dc: 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 . operator
    rator didnt work so i need to use ->
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x0000555555552f7 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 . operator
    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  0x0000555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...},
    node_num=3) at buggy.cpp:25
#1  0x000055555555407 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:

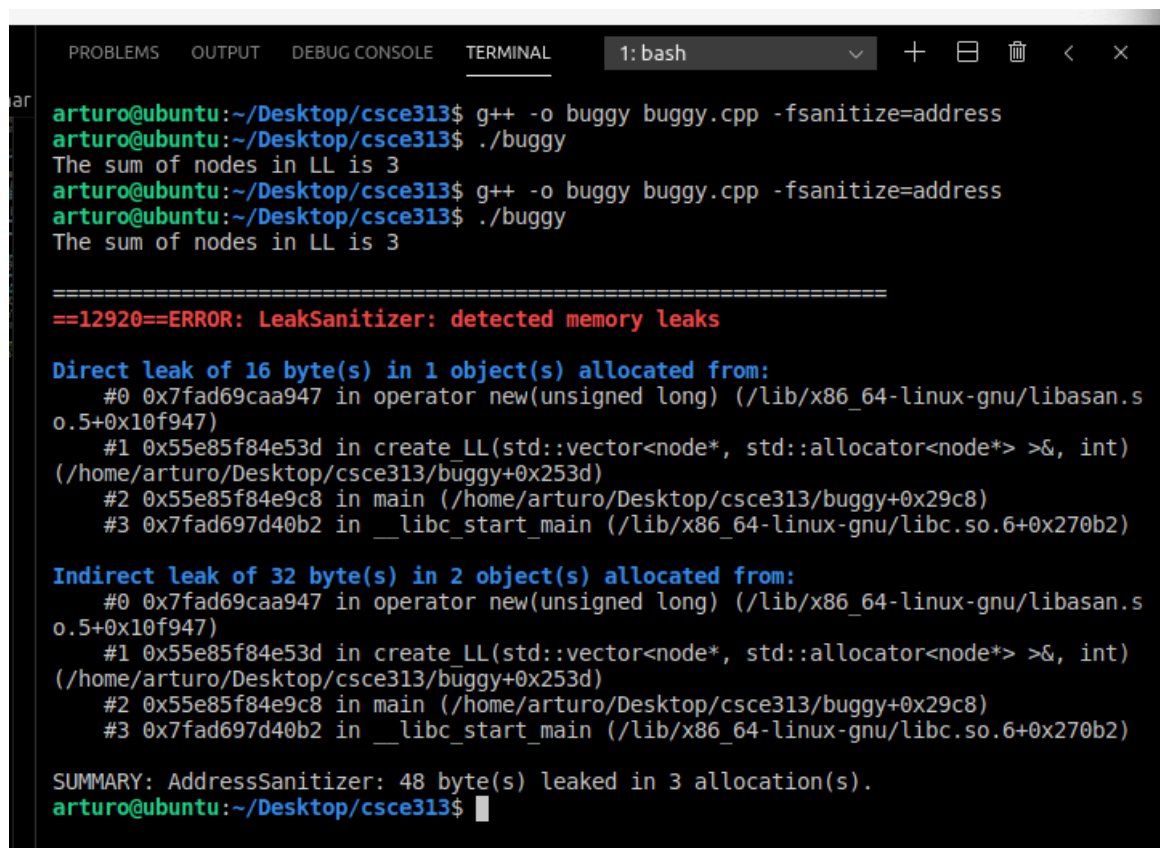
```
(gdb) n
30      for (int i = 0; i < node_num; i++) {
(gdb) print mylist
$2 = std::vector of length 3, capacity 3 = {0x55555556bed0,
      0x55555556bef0, 0x55555556bf10}
(gdb) █

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 were null. This is all good but if we run the program in gdb fresh I will get a segmentation fault 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 without segmentation faults.

<pre>buggy.cpp > create_LL(vector<node*>&, int) 17 18 void create_LL(vector<node*>& mylist, int node_num) { 19 mylist.assign(node_num, NULL); /// all elements are null 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 node 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;</pre>	<pre>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/licenses/gpl.html> This is free software: you are free to change and redistribute it. 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 The sum of nodes in LL is 3 [Inferior 1 (process 12673) exited normally] (gdb) █</pre>
---	--

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



```
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.so.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.so.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_NUM = 3;
    std::vector<node*> mylist;
    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];
    }
}

```

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\$

