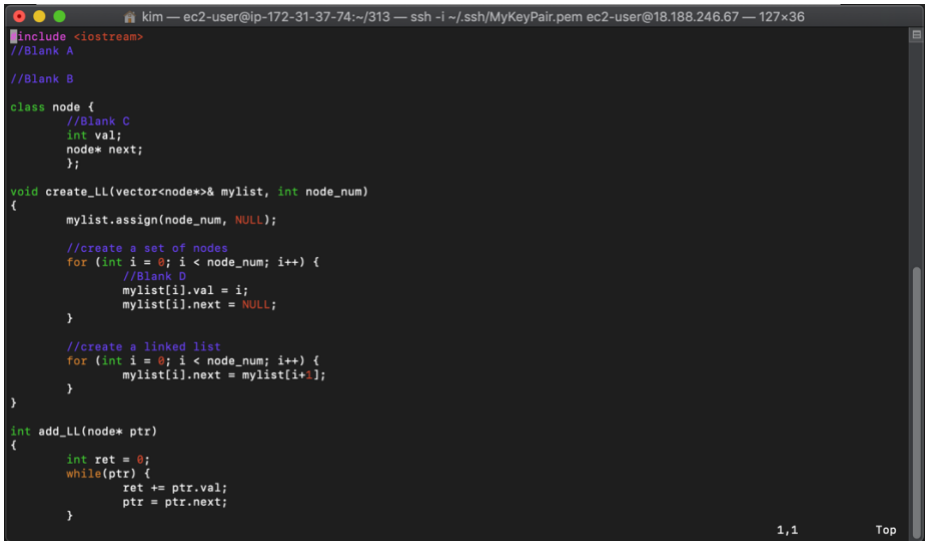
Kim Nguyen

CSCE 313-502

August 29, 2019

Report for PA0

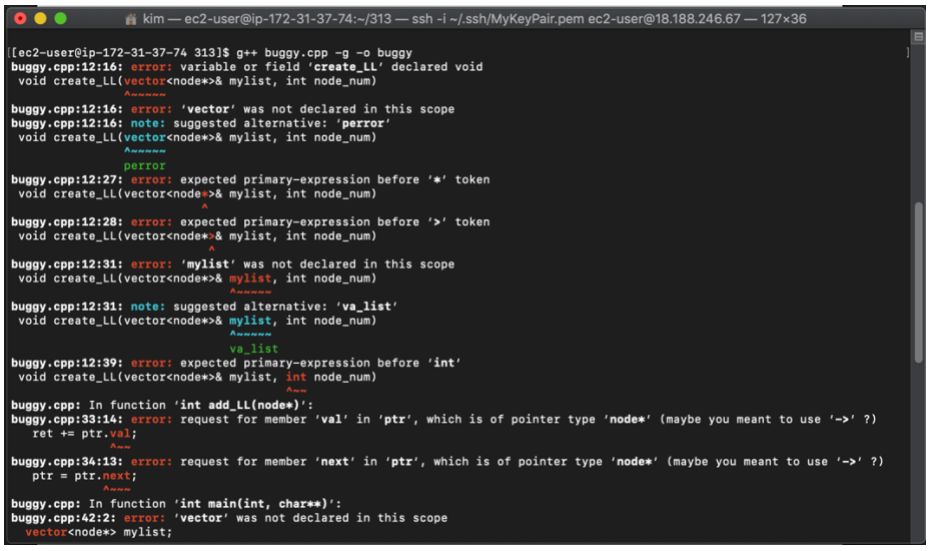
1. System Setup

- 

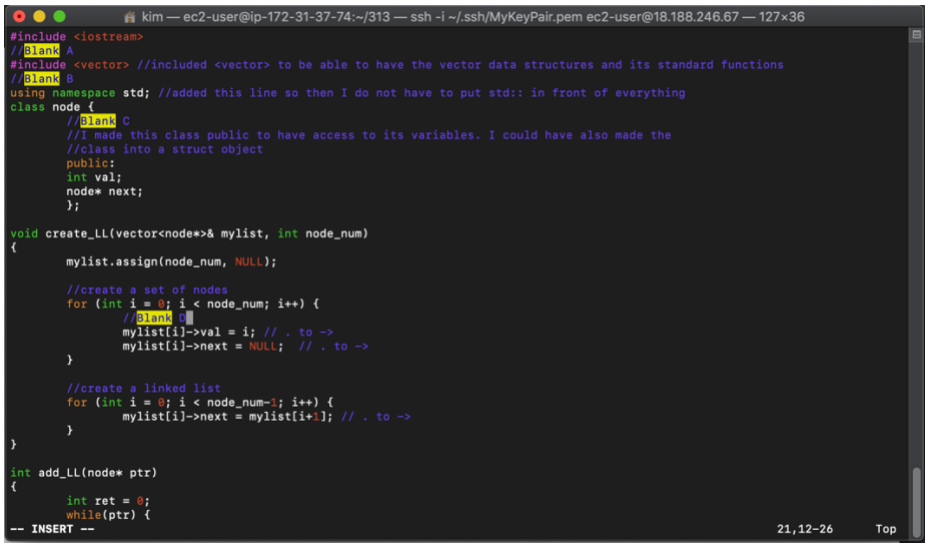
- I used AWS EC2 to set up a Linux virtual machine.

2. C++ Compilation

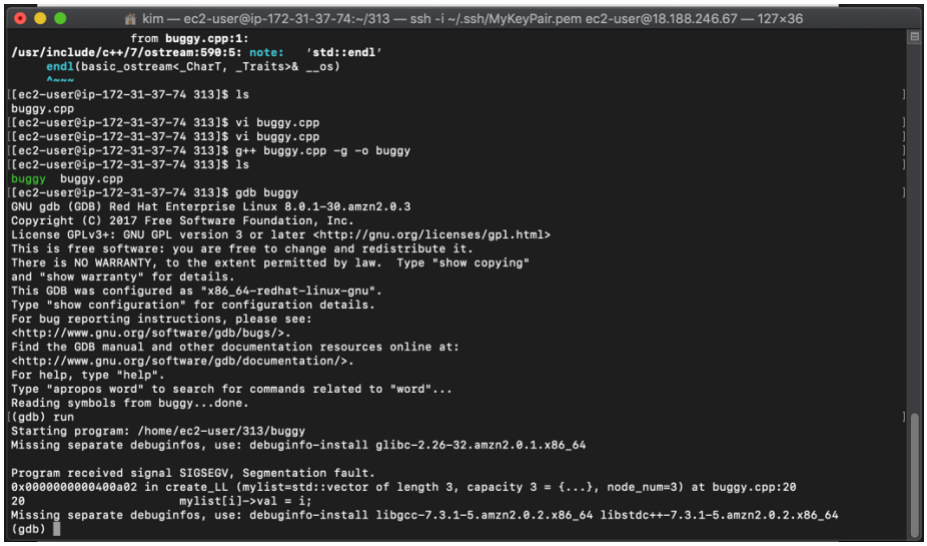
- For the command on how to compile the buggy.cpp file, I used g++ and -g for gdb and -o to make a Unix executable file. When I tried to compile

- 

3. Compile Time Error Debugging

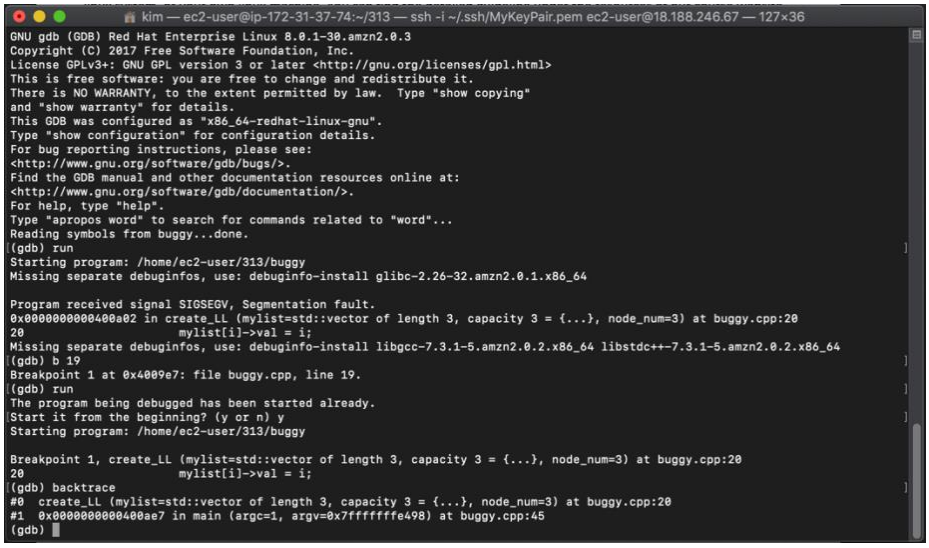
    - I have made notes of my changes in the code's comments.
    - 

4. C++ Program Execution

    - I encountered a segmentation fault at line 20 and then the program stopped running.
    - 

5. C++ Compilation in Debug Mode

    - The command I used was **b** using gdb in order to set a breakpoint and inserted an integer to signify which line of code I wanted to set a breakpoint in.
    - To start running the program I used the **run** command and the **step** command whenever I am jumping through breakpoints.

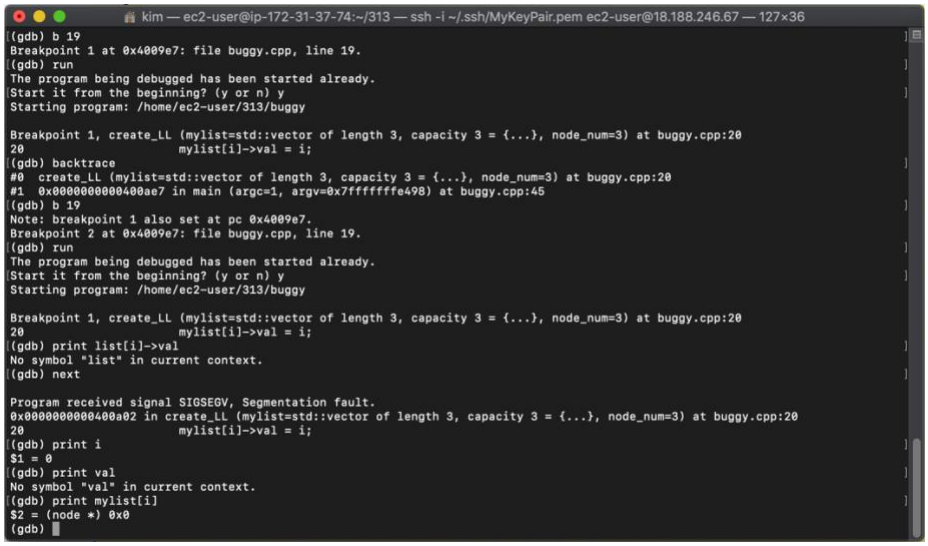
- 

6. GDB Start/Run/Backtrace

- 

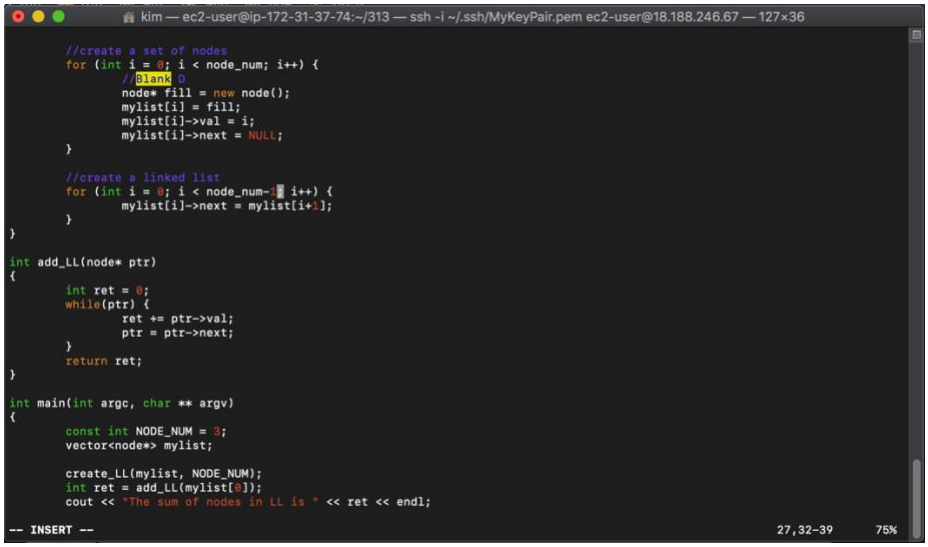
7. GDB Breakpoint/Print

- I added a breakpoint to line 19 and used the **print** command in order to print the value of **i** and **mylist[i]**.



8. C++ Runtime Error Fix (Null-Pointer)

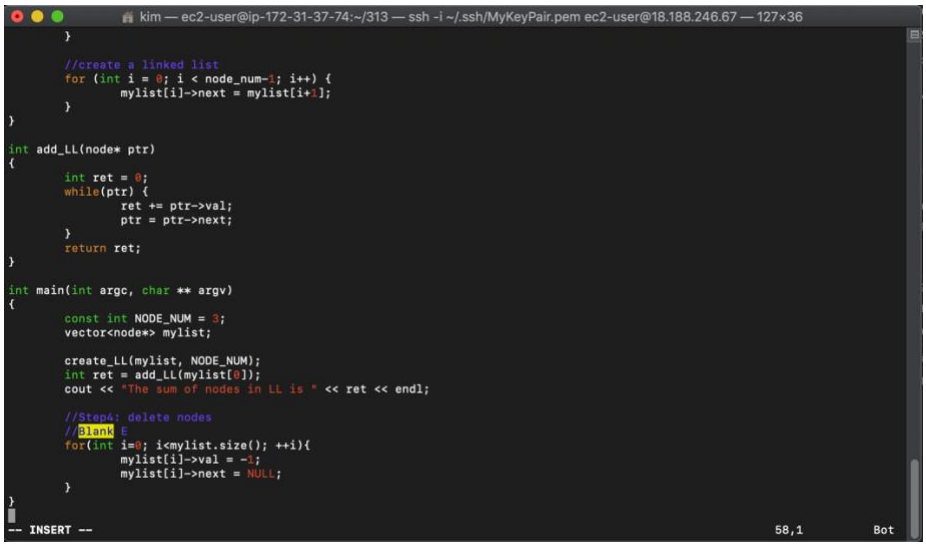
- Under **//Blank D** I made a new variable, **fill**, that is a dynamically empty node. I used this to initialize the current element of the vector in order to prevent the error that was occurring. What was happening was that there was nothing inside the vector and the program would error, so initializing the vector element prevented this.



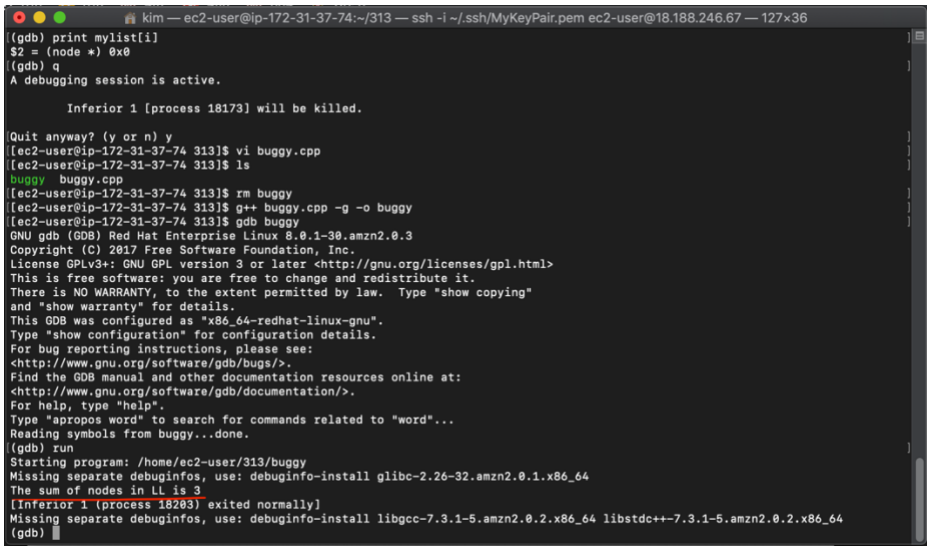
9. C++ Runtime Error Fix (non-NULL garbage value Pointer)

- In the second for loop of the **create_LL** function, the error occurred because of the **ptr->val** in **add_LL**. To stop this error from happening, I subtracted one from **node_num** in **create_LL**'s second for loop in order to prevent the program from going out of bounds.
- 

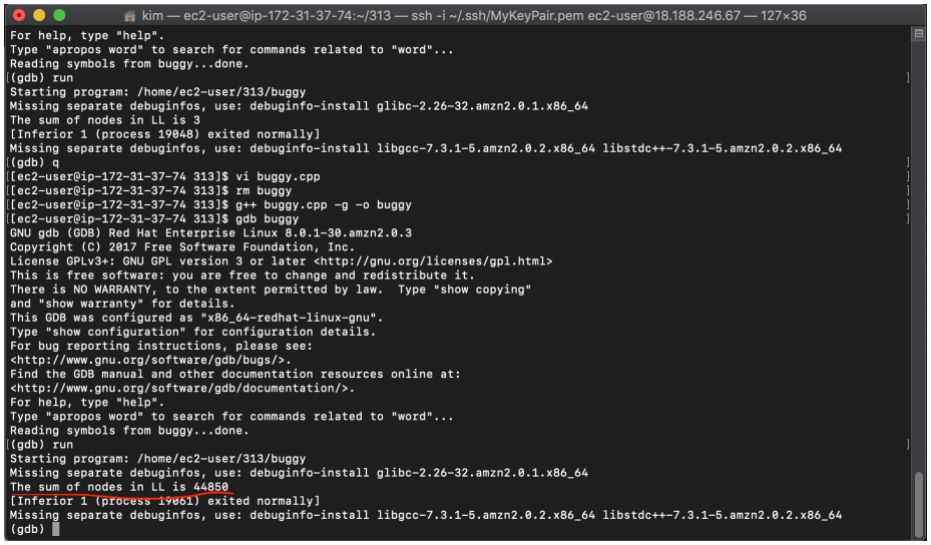
10. Dynamically Allocated Memory Deletion

- Under **//Blank E** I coded a for loop that will make all of the **val**'s -1 and the next to be **NULL** to prevent a memory leak from happening.
- 

In the end the changes I made to the code allowed it to start working. I have tested this by changing the value of NODE_NUM in the main function from 3 to several different values. I used the triangular sum formula to ensure that the program is computing the correct number every time.



This was when NODE_NUM = 3.



This was when NODE_NUM = 300.

~~Proof: 300*(300+1)/2 = 44850~~

Proof: 299*(299-1)/2 = 44850 (We do NODE_NUM-1 as the variable because it starts as 0+1+... instead of 1+2+...)