CSCE 313
PA0 Report
Jeffrey Xu
08/28/20

1 Compilation

To begin debugging, the program was compiled and the error messages were analyzed. First, the vector package wasn't included and the namespace for the program wasn't specified. Therefore, by including the vector package from std and specifying the namespace to std these errors went away.

Next, there were issues with referencing data values inside the node class. This was due to the fact that the vector for *mylist* holded a vector of node pointers and not node objects. The code was trying to access data members without dereferencing the pointer so the arrow operator was used instead of directly trying to access the data values.

The next error message stated that the data members inside of node were declared as private and couldn't be accessed outside of the class. This was a very simple fix as declaring all of the data members as public allowed for access to the data members outside of the class.

This wasn't really a compilation error, but the code also didn't delete the nodes when it finished, so at the end of the main function, the nodes in the vector were properly deallocated.

2 Runtime

After the program was able to compile, an executible file was created and the program was executed on $g++$. A segmentation fault occurred somewhere in the program.

2.1 GDB

GDB was used to help debug the program. By backtracing through the program in gdb, it was noticed that the program hits a segmentation fault at line 15. This error occurred because no node object was ever created for each index in the vector, therefore by trying to access the data member *val* inside of the *ith* index of the vector, the program was trying to access data that didn't existed, hence the segmentation fault. This was easily fixed by allocating memory to hold a node object in each index of the vector.

Running the program in gdb again, a segmentation fault still occurs. By using backtracing, it was found that the issue was occuring in the second part of the *create_LL* function when the function is assigning the *next* data member of each node. The loop went out of range as it was trying to access an index that wasn't in the range of the size of the vector. This was easily fixed by reducing the range that the loop goes through. Instead of going all the way through *node_num*, the loop iterated to *node_num* -1.

2.2 AddressSanitizer

Using AddressSanitizer was similar to gdb but had less functionality. Initially, the error messages were the same, a segmentation fault on a null address when trying to access data members in *create_LL*. This was fixed by allocating memory to each index of the vector.

The compiler still reported errors however. The program was still aborted due to a buggy address. This was debugged to be found to be due to the second part of *create_LL* as it was trying to access a memory location that was outside of the range of the vector. This was solved by changing the range of which the loop iterated through to one less than the original value.