// Keaton Spiller

// CS302

// 02/19/2024 Winter 2024

// Program # 2

Debugger Writeup

One thing I used the debugger for was understanding why manually calling the delete\_all function to deallocate dynamic memory in the Double Linked List would give a “free(): double free detected” error. This means I was deleting the same memory multiple times. Yet, all the dynamic memory was freed, and I had no memory leaks.

I didn’t know the ramifications of deleting multiple times. When the program ends without manually calling the delete all function in the menu and instead is called in the deconstructor of the Double linked list at the end of the it’s lifespan, the program has no problem not deleting multiple times.

Using gdb I was able to figure out that any string converted into a char \* with c\_str() may give string class access to deallocation, and I was over deleting because I was deallocating the char \* delete [], at the end of deconstructor when the string class was also deleting.

I would like to use gdb more to trace through the logic as objects as passed up through the hierarchy and be able to visualize the “is a” and “containing” relationships as data flows through the architecture. This is especially important when we get to virtual dynamic classes because then these skills will help me understand where the code is actually going instead of where I think it’s going.