Program2

Ching-Wei Lin

#987563037

Debugger:

This program is more about the dynamic binding which is more about dealing with the classes and about how to make the classes connected and support each other. Dynamic binding is really convenient that we could declare the data type of the object at the run time. Therefore, the data type part is like just reviewing the past study. In double linked list and circular linked list, they both have several pointers, which segmentation fault happens all the time. I used to gdb debugger to find my segmentation fault in the remove all function. First, I found that I have seg fault at the moment I closed the program, but not always. Then, I just left it away. When I came to build my remove all function, I used the same wrapper function for the destructor because in some way, destructor is the same as remove all. I found every time I call the remove all function, seg fault came out. Thus, I used gdb to search my seg fault. Finally, I realized I was calling a head arrow previous which the head was pointing to null at the end of remove all function. It happened only when I inserted data before. That’s why it didn’t happen sometimes.

Analysis:

When I started to work on this assignment, I do not quite understand how dynamic binding works. I just knew the concept that we declare the data type at the run time. While I was working on it, I try many syntaxes to make it work. For example, three kinds of material have their own feature, so I planned to ask the user which type of material does he want to add in the main function. Then, set it as a condition and lead to the corresponding section which asks different questions to fulfill the corresponding data, like reading has six data member but lab has just four. All the works above are in main function, so I had to call the add function with different argument for different data type. Then, I found it is not the spirit of dynamic function. It is convenient at all. I rewrote my code and hide all the local variables in the member function. As a result, all of my add function, display function, remove function and remove all function are all the same for each data type. I make the abstract for the base class and just implement the derived classes. In the main function, I only have to call the function without thinking what type is that material because the program will make it all set for me automatically and I think that is the spirit of dynamic binding. If I have more time to finish the program, I will add more functions in the exercise class which can manage the circular linked list in it. Or maybe I will try another dynamic binding in it. So that will be two layers dynamic binding, which would be very cool. However, I think double linked list is not the best data structure for this program because we are not doing backward traverse. Using double linked list would only use more memory and move more steps for setting the previous pointer which also lowers the efficiency. Similar thought, we do not need circular linked list f or the exercise material. Using circular linked list is not making the program more efficiently because we don’t know where is the data we are looking for. Therefore, no matter retrieving, adding or removing I search the list from the head. It is meaningless to start from the end. While, it lowers the efficiency just a little.