COP3530 17F Project 1

Harsh Patel

8011-0499

23HE

By submitting this document, I affirm that all the work submitted is my solely my own and that in completing this project I did nothing contrary to either the spirit or the letter of the UF Honor Code

Section I

Implementation Summary

	SSLL	PSLL	SDAL	CDAL	CBL
My template class has the required name, file extension, and template parameters. **[y/n]**	y	y	y	y	y
My template class was implemented *efficiently* using the required technique (*e.g.*, linked list with a pool of free nodes) as described in Part I and as amended in the announcements. **[y/n]**	y	y	y	y	y
I have implemented and throughly tested each of the methods described in Part I and they ***all*** behave correctly. **[y/n]**	y	y	y	y	y
I have implemented and throughly tested (on one of the official test machines) each of the *big five* member functions and they ***all*** behave correctly. **[y/n]**	n	n	n	n	n
I have implemented and throughly tested each of the *type members* as described in Part II. **[y/n]**	y	y	y	y	y
I have implemented and throughly tested *efficient* iterators (both **const** and non-const) over a list instance's data and my list supports the iterator creation operations as described in Part II. **[y/n]**	y	y	y	y	y
I have verified (by testing) that a **const** instance of my list class and the data it holds cannot be modified, either through the list operations nor via iterator over the list's elements. **[y/n]**	y	y	y	y	y
I wrote my tests using CATCH (not required, *except* for the CBL class tests). **[y/n]**	n	n	n	n	n
I have verified my template class is memory-leak free using valgrind. **[y/n]**	n	n	n	n	n
I certify that all of the responses I have given for this list class are TRUE [*HP*]	y	y	y	y	y

Section II

Entering this project, I briefly understood what a link list was, but did not know how to properly implement them; I learned how to effectively use pointers, references, and dereferences. Now my project is not perfect, and I understand that. I tried day and night on fixing some of the memory leaks I was getting and just overall bugs in my code, but I could not figure out all of them. Upon running my program, I had 20 total memory leaks in project 1. I tried my best to debug those memory leaks, but could only reduce them down to 13. I can proudly say that I have grown as a programmer because I have learned how to locate certain memory leaks and resolve them. Also, I did not know what copy and move constructors were until you announced we had to implement them in our project. I had to do a lot of reading on them and even then, I do not know if I did them correctly. On the bright side, I understand the purpose behind them and why we need them in our classes. By far the best learning experience in this project was creating the CDAL, and I am really proud of it, even though there are some memory leaks in it. It took me two weeks to property conceptualize and program it. It was not an easy class to create and it ate up all my time. On the other hand, the CBL was the most fun class to create because of its circular nature. At first it took a while to abstract it, but after that the code came out naturally. I am glad you gave this project because it just solidified my understanding of linked list and the different ways they can be abstracted and implemented. The only thing I regret about this project is not doing the CATCH for CBL because I did not have enough time with all my other classes hammering me down with assignments. Truly sorry about this.

Section III

COMMAND TO RUN:
g++ -o play SSLL_test.cpp
g++ -o play PSLL_test.cpp
g++ -o play SDAL_test.cpp
g++ -o play CBL_test.cpp
g++ -o play CDAL_test.cpp

```
PSLL STARTS HERE
--------------------
1
1234
543
24
254
244
5
76543
56
100
1
100
[1, 1234, 543, 24, 254, 244, 5, 76543, 56, 100]
1
1[1234, 543, 24, 254, 244, 5, 76543, 56, 100]Program ended with exit code: 0
```

```
SSLL STARTS HERE
--------------------
1
1234
543
24
254
244
5
76543
56
100
1
100
[1, 1234, 543, 24, 254, 244, 5, 76543, 56, 100]
1
1[1234, 543, 24, 254, 244, 5, 76543, 56, 100]Program ended with exit code: 0
```

```
SDAL STARTS HERE
--------------------
The list was empty so this (24) will be your first element
1
1234
543
24
254
244
5
76543
56
100
1
100
[1, 1234, 543, 24, 254, 244, 5, 76543, 56, 100]
Lenght is: 10

1
1[1234, 543, 24, 254, 244, 5, 76543, 56, 100]
Lenght is: 9
Program ended with exit code: 0
```

```
CDAL STARTS HERE
--------------------
1
1234
543
24
254
244
5
76543
56
100
1
100
[1, 1234, 543, 24, 254, 244, 5, 76543, 56, 100]
1
it comes here 2
1[1234, 543, 24, 254, 244, 5, 76543, 56, 100]Program ended with exit code: 0
```

```
CBL STARTS HERE
--------------------
The list was empty so 24 will be your first elemenet!
1
1234
543
24
254
244
5
76543
56
100
1
100
Tail: 9
Head: 49
50
[1, 1234, 543, 24, 254, 244, 5, 76543, 56, 100]
Head is: 49
Tail is: 9

1
1Tail: 8
Head: 49
50
[1234, 543, 24, 254, 244, 5, 76543, 56, 100]
Head is: 49
Tail is: 8
Program ended with exit code: 0
```

This is the proof that I used valgrind to test it, but couldn't resolve all memory leaks. This test is all the test files in one file for ease of convenience.