Title: COP3530 17F Project 2

Mingyue Chen

63238271

Section #: 1087

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: insert the completed checklist table here

Implementation Summary

	BST LE AF	BST R OOT	BST R AND	AVL	HASH OPEN	HASH BUCKE T
My template class has the required name, file extension, and template parameters. **[y/n]**	y	y	y	y	n	n
My template class was implemented *efficiently* using the required technique as described in the project description. **[y/n]**	y	y	y	y	n	n
I have implemented and throughly tested each of the **required** methods and they ***all*** behave correctly. **[y/n]**	n	n	n	n	n	n
AVL-based map only: I have throughly tested and verified that at the completion of an insertion/removal that the tree and all of its subtrees are AVL balanced.	—	—	—	n	—	—
I have implemented and throughly tested each of the **Group 1** methods and they ***all*** behave correctly (bonus). **[y/n]**	n	n	n	n	n	n
I have implemented and throughly tested *efficient* iterators (both **const** and non-const) over a map instance's data and my map supports the bonus **Group 2** methods iterator creation operations (bonus). **[y/n]**	n	n	n	n	n	n
I have verified by testing that a **const** instance of my map class and the data it holds cannot be modified, either through the map operations nor via iterator over the map's	n	n	n	n	n	n
I wrote my tests using CATCH. **[y/n]**	y	y	y	y	n	n
I have verified my map class is memory-leak free using valgrind. **[y/n]**	y	y	y	y	n	n
I certify that all of the responses I have given for this list class are TRUE [*your*	MC	MC	MC	MC	MC	MC

Section II: Learning experience

I think the most difficult part is to make sure AVL tree is always AVL balanced. I think the easiest part is to lookup function for all the classes. I understand how to insert an element to an AVL tree and how to delete elements properly. I also learned to generate a random number correctly. I understand how to rotate trees as well.

Section III: Testing

BSTLEAF:

Command line:

```
g++ -std=c++11 main2.cpp -o p && ./p
```

Outputs

```
===============================================================================

All tests passed (12 assertions in 1 test case)
```

Memory leak:

```
valgrind --tool=memcheck --leak-check=full ./main2.cpp

==24059== Memcheck, a memory error detector

==24059== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.

==24059== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info

==24059== Command: ./main2.cpp

==24059==

./main2.cpp: 8: ./main2.cpp: using: not found

: not found: 8: ./main2.cpp:

./main2.cpp: 9: ./main2.cpp: Syntax error: "(" unexpected

==24059==

==24059== HEAP SUMMARY:

==24059==     in use at exit: 2,789 bytes in 88 blocks

==24059==   total heap usage: 96 allocs, 8 frees, 5,893 bytes allocated

==24059==

==24059== LEAK SUMMARY:
```

```
==24059==    definitely lost: 0 bytes in 0 blocks

==24059==    indirectly lost: 0 bytes in 0 blocks

==24059==      possibly lost: 0 bytes in 0 blocks

==24059==    still reachable: 2,789 bytes in 88 blocks

==24059==         suppressed: 0 bytes in 0 blocks

==24059== Reachable blocks (those to which a pointer was found) are
not shown.

==24059== To see them, rerun with: --leak-check=full --show-leak-
kinds=all

==24059==

==24059== For counts of detected and suppressed errors, rerun with: -v

==24059== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```

BSTROOT:

Command line:

g++ -std=c++11 main2.cpp -o p && ./p

Outputs

```
======================================================================
=========

All tests passed (12 assertions in 1 test case)
```

Memory leak:

```
valgrind --tool=memcheck --leak-check=full ./main2.cpp

==24059== Memcheck, a memory error detector

==24059== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et
al.

==24059== Using Valgrind-3.11.0 and LibVEX; rerun with -h for
copyright info

==24059== Command: ./main2.cpp

==24059==
./main2.cpp: 8: ./main2.cpp: using: not found

: not found: 8: ./main2.cpp:

./main2.cpp: 9: ./main2.cpp: Syntax error: "(" unexpected

==24059==

==24059== HEAP SUMMARY:

==24059==     in use at exit: 2,789 bytes in 88 blocks

==24059==   total heap usage: 96 allocs, 8 frees, 5,893 bytes
allocated

==24059==

==24059== LEAK SUMMARY:

==24059==    definitely lost: 0 bytes in 0 blocks

==24059==    indirectly lost: 0 bytes in 0 blocks

==24059==      possibly lost: 0 bytes in 0 blocks

==24059==    still reachable: 2,789 bytes in 88 blocks

==24059==         suppressed: 0 bytes in 0 blocks

==24059== Reachable blocks (those to which a pointer was found) are
not shown.

==24059== To see them, rerun with: --leak-check=full --show-leak-
kinds=all

==24059==

==24059== For counts of detected and suppressed errors, rerun with: -v

==24059== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```

BSTRAND:

Command line:

```
g++ -std=c++11 main2.cpp -o p && ./p
```

Outputs

```
===============================================================================
=========

All tests passed (12 assertions in 1 test case)
```

Memory leak:

```
valgrind --tool=memcheck --leak-check=full ./main2.cpp

==24059== Memcheck, a memory error detector

==24059== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et
al.

==24059== Using Valgrind-3.11.0 and LibVEX; rerun with -h for
copyright info

==24059== Command: ./main2.cpp

==24059==

./main2.cpp: 8: ./main2.cpp: using: not found

: not found: 8: ./main2.cpp:

./main2.cpp: 9: ./main2.cpp: Syntax error: "(" unexpected

==24059==

==24059== HEAP SUMMARY:

==24059==     in use at exit: 2,789 bytes in 88 blocks

==24059==   total heap usage: 96 allocs, 8 frees, 5,893 bytes
allocated

==24059==

==24059== LEAK SUMMARY:

==24059==    definitely lost: 0 bytes in 0 blocks

==24059==    indirectly lost: 0 bytes in 0 blocks

==24059==      possibly lost: 0 bytes in 0 blocks

==24059==    still reachable: 2,789 bytes in 88 blocks

==24059==         suppressed: 0 bytes in 0 blocks

==24059== Reachable blocks (those to which a pointer was found) are
not shown.
```

```
==24059== To see them, rerun with: --leak-check=full --show-leak-
kinds=all

==24059==

==24059== For counts of detected and suppressed errors, rerun with: -v

==24059== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```

AVL:

Command line:

g++ -std=c++11 main2.cpp -o p && ./p

Outputs

```
========================================================================
=========

All tests passed (12 assertions in 1 test case)
```

Memory leak:

```
valgrind --tool=memcheck --leak-check=full ./main2.cpp

==24059== Memcheck, a memory error detector

==24059== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et
al.

==24059== Using Valgrind-3.11.0 and LibVEX; rerun with -h for
copyright info

==24059== Command: ./main2.cpp

==24059==

./main2.cpp: 8: ./main2.cpp: using: not found

: not found: 8: ./main2.cpp:

./main2.cpp: 9: ./main2.cpp: Syntax error: "(" unexpected

==24059==

==24059== HEAP SUMMARY:

==24059==     in use at exit: 2,789 bytes in 88 blocks

==24059==   total heap usage: 96 allocs, 8 frees, 5,893 bytes
allocated
```

```
==24059==
==24059== LEAK SUMMARY:
==24059==    definitely lost: 0 bytes in 0 blocks
==24059==    indirectly lost: 0 bytes in 0 blocks
==24059==      possibly lost: 0 bytes in 0 blocks
==24059==    still reachable: 2,789 bytes in 88 blocks
==24059==         suppressed: 0 bytes in 0 blocks
==24059== Reachable blocks (those to which a pointer was found) are
not shown.
==24059== To see them, rerun with: --leak-check=full --show-leak-
kinds=all
==24059==
==24059== For counts of detected and suppressed errors, rerun with: -v
==24059== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from
0)
```