Test Plan For MP5

Student: Yulin Xiao
CUID: C16278133
Student UserName: Yulinx

TEST Plan:(See MP5 lab5.c) Expected Results:(see Test Log For MP5)

1. Do the unit driver with custom test 0 through 6 (4-6 is my added binary search tree tests) to verify that my code is able to do the basic binary search tree operation (insert, remove):

```
lab5 -u 0
//test to remove leaves, 12 and 20, then internal nodes
// 8, 24, 40 with one child, then 16, 48 with two children
lab5 -u 1
// tests: (48) is missing its right-left child and
//         (16) is missing its left-right child
lab5 -u 2
// deletion with many children
lab5 -u 3
// check replace for duplicate key
lab5 -u 4
// check replace and double deletion for duplicate key
lab5 -u 5
// complete deletion, first for parent(100,85(L),65(R),200(Root)) and then
for child(67,68,66)
lab5 -u 6
// complete deletion, remove root
```

Here is my stack, the unit test driver that I wrote:

```
if (UnitNumber == 4)
{
 // check replace and duble deletion for duplicate key
 const int ins[] = {10, 10};
 const int del[] = {10, 10};
              unitDriver(ins, sizeof ins / sizeof(int),
             del, sizeof del / sizeof(int));
}
if (UnitNumber == 5)
{
 // complete deletion, first for parent and then for child
 const int ins[] = {200,100,50,150,25,75,125,175,65,85,135,80,130,140,78,82,67,66,68};
 const int del[] = {100,85,125,150,50,82,80,78,65,67,68,66,175,130,135,75,140,200,25};
  unitDriver(ins, sizeof ins / sizeof(int),
 del, sizeof del / sizeof(int));
}
```

```
if (UnitNumber == 6)
{
// complete deletion, remove root
const int ins[] = {200,100,50,150,25,75,125,175,65,85,135,80,130,140,78,82};
const int del[] = {200,100,125,130,135,140,150,175,50,65,75,78,80,82,85,25};
 unitDriver(ins, sizeof ins / sizeof(int),
 del, sizeof del / sizeof(int));
 }
```

2. Do the unit driver 0 and 1 with ./lab5 -v -f avl to test my AVL tree insertion (the deletion is not completed, but the deletion do not violate the AVL property, so it do not generate the assertion fault)
3. Do the command line arguments of:

```
lab5 -o -w 5 -t 0 -v
// tests inserts only and prints tree
lab5 -r -w 5 -t 0 -v -s 1
// same with random tree
lab5 -p -w 5 -t 0 -v -s 2
// same with random tree
```

   To validate that the tree remains in binary search tree's property
4. Do the command line arguments of:

```
lab5 -o -w 20 -t 1000000
// tests inserts and accesses
lab5 -r -w 20 -t 1000000
// same with random tree
lab5 -p -w 20 -t 1000000
// same with poor insertion order
```

   To verify that the expected number of searches predicted by the theory matches the measured performance from my program to three significant digits when run with 1,000,000 trials.
5. Do the performance evaluations!
   for the worst insertion order, the expected time complexity is O(n).

Expected Results:(see Test Log For MP5)

Drawback: when I do "./lab5 -f avl -p -w 20 -t 1,000,000"

that is insert avl tree for poor insertion policy and it took too long to generate results. so I do: "./lab5 -f avl -p -w 10 -t 1,000"