Test Log For MP5

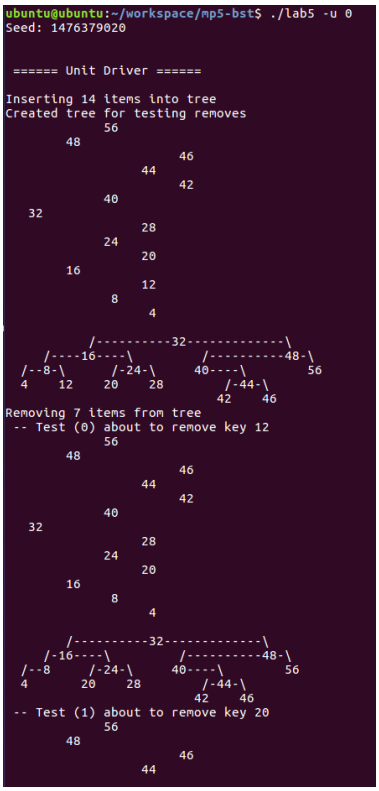
Student: Yulin Xiao

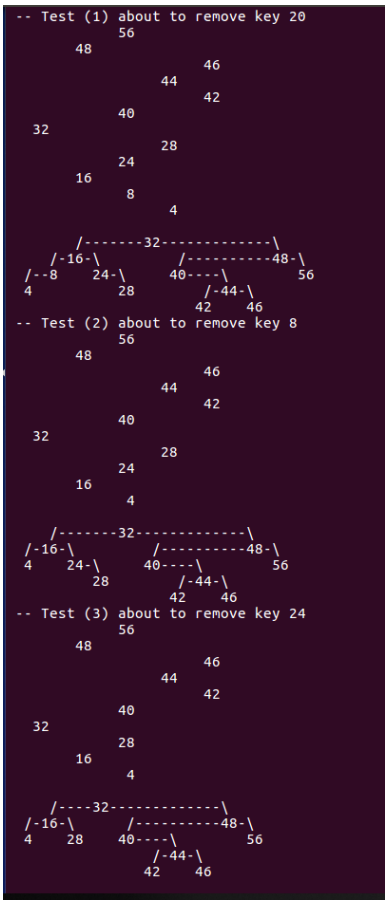
CUID: C16278133

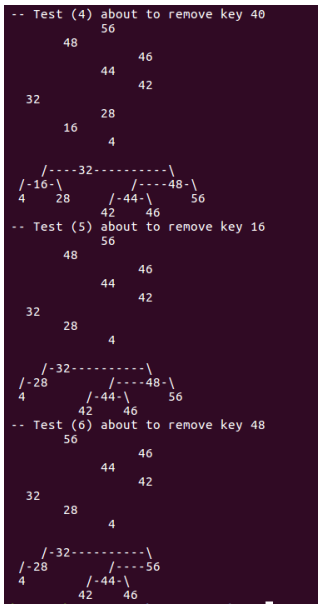
Student UserName: Yulinx

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
```



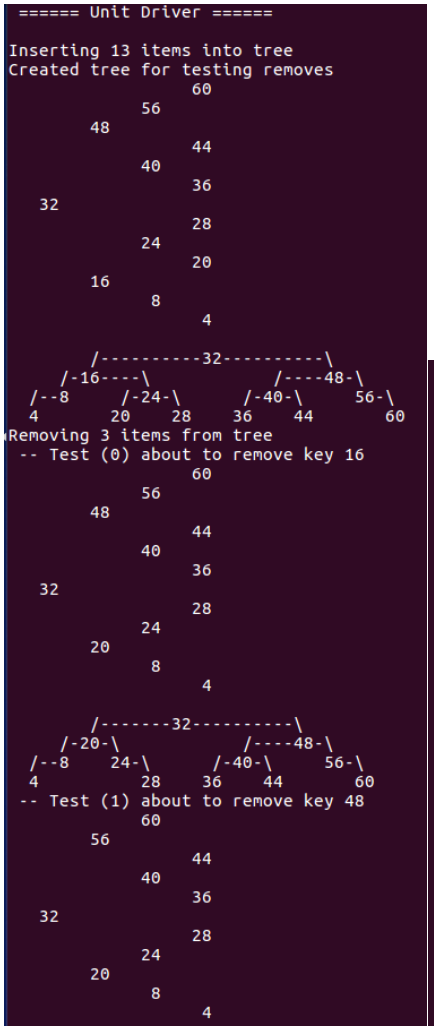


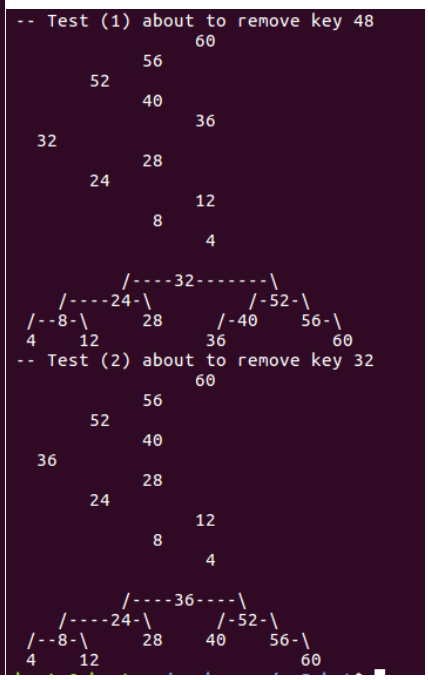


lab5 -u 1

// tests: (48) is missing its right-left child and

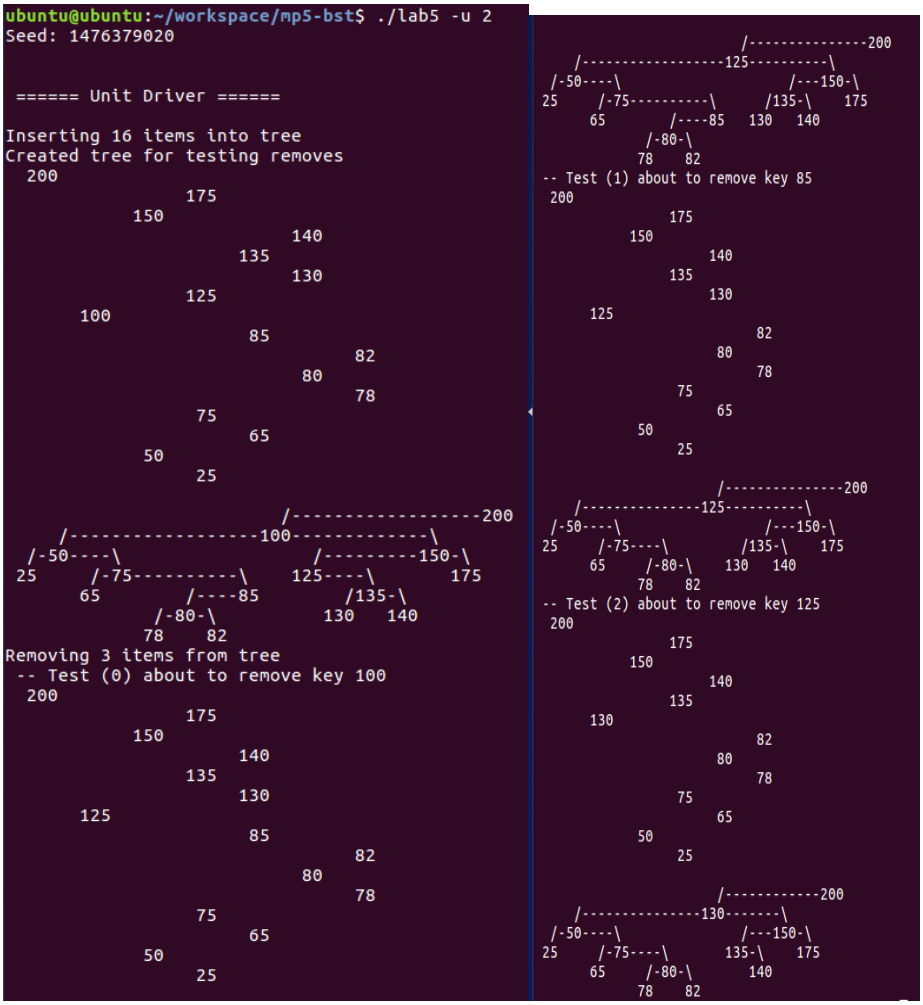
// (16) is missing its left-right child





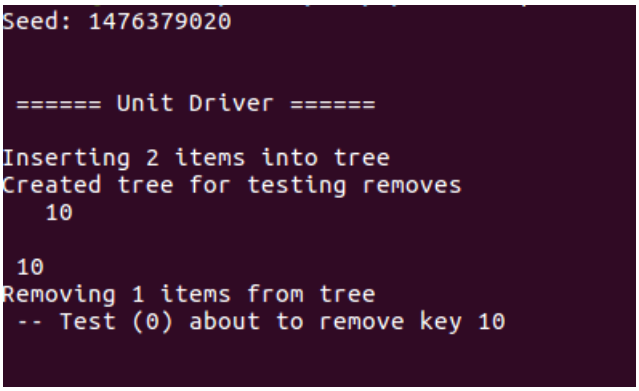
lab5 -u 2

// deletion with many children



lab5 -u 3

// check replace for duplicate key

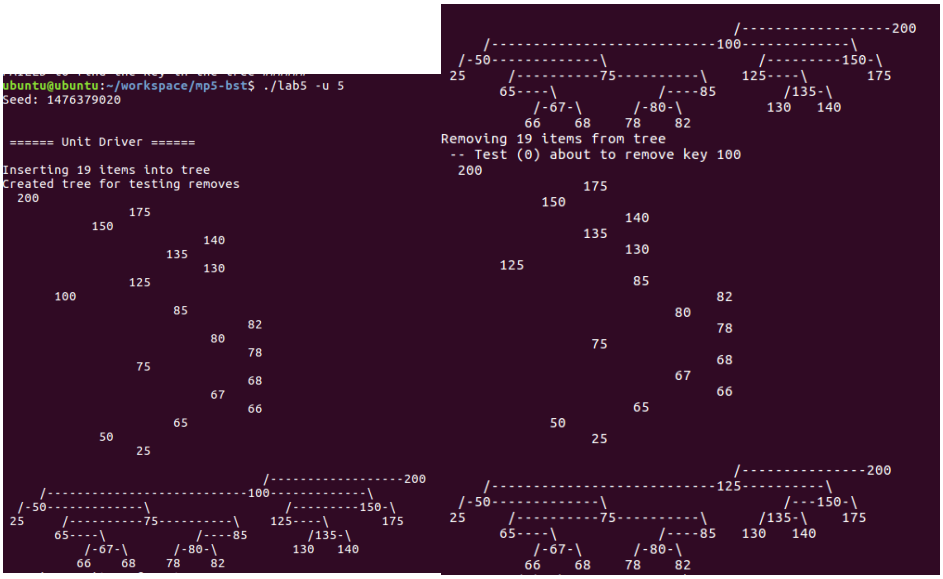


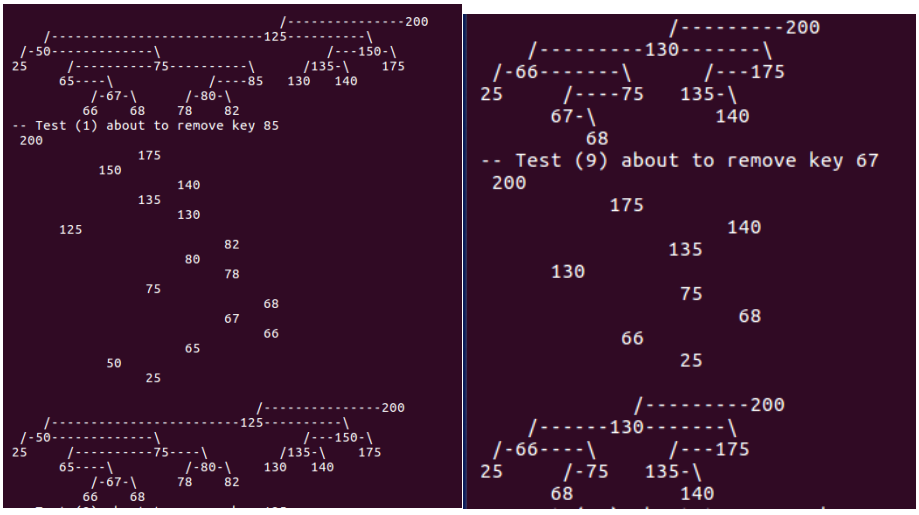
lab5 -u 4

// check replace and double deletion for duplicate key

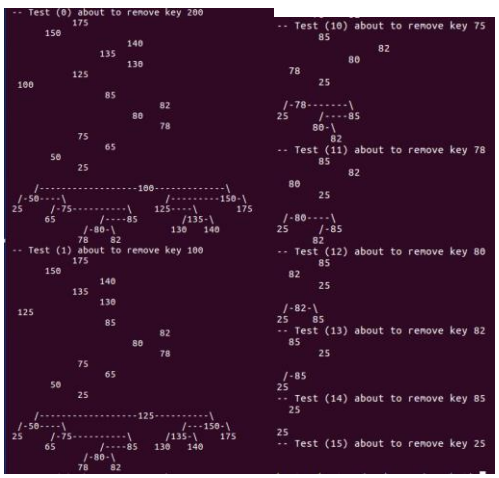
lab5 -u 5

// complete deletion, first for parent(100,85(L),65(R),200(Root)) and then for child(67,68,66)

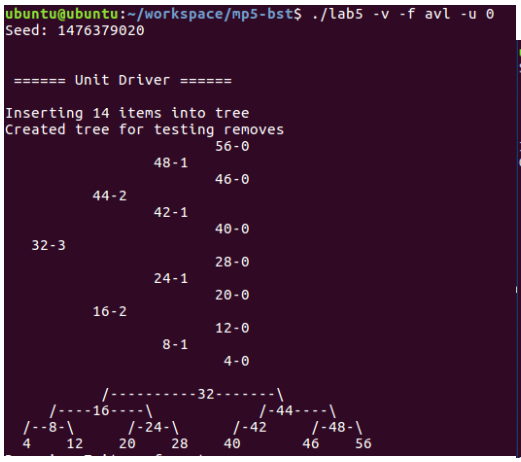


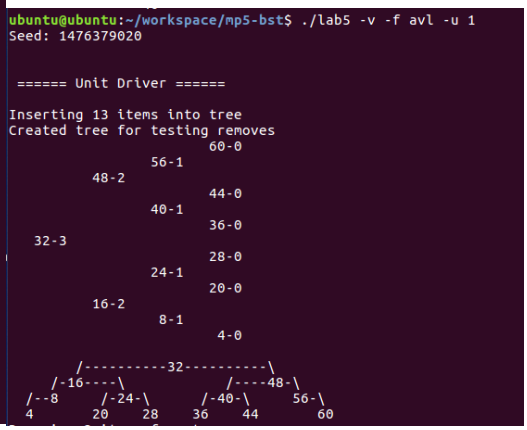


lab5 -u 6 // complete deletion, remove root



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) (Check the property)

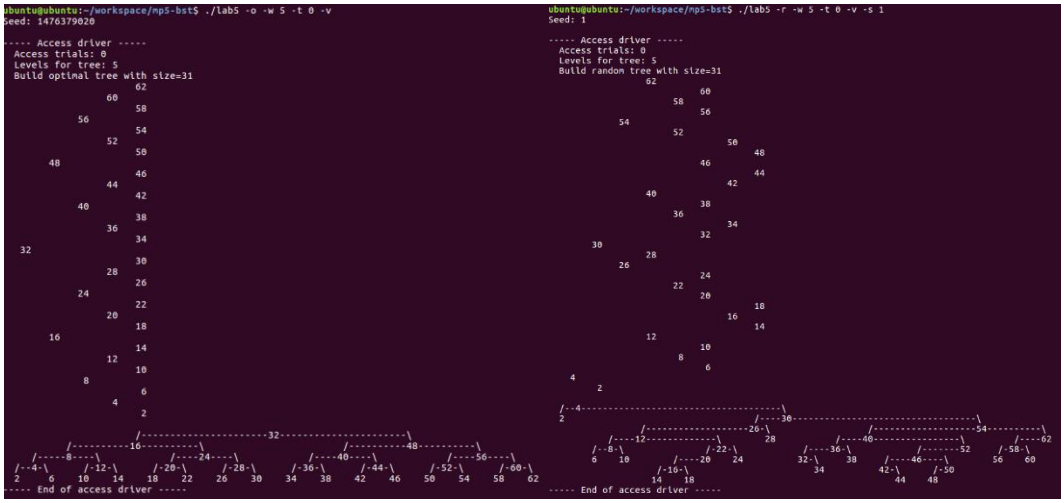


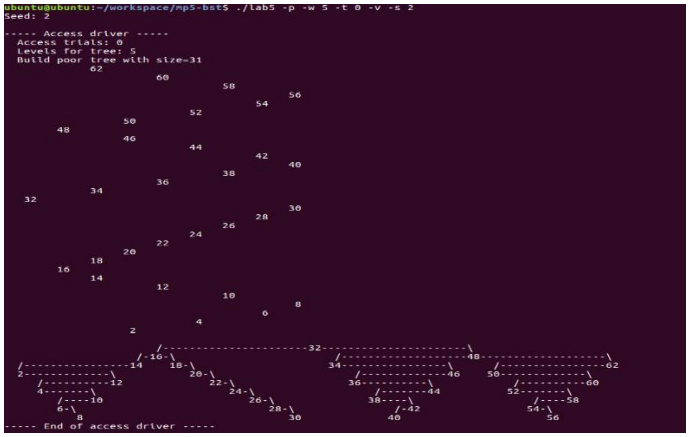


3. Do the command line arguments of:

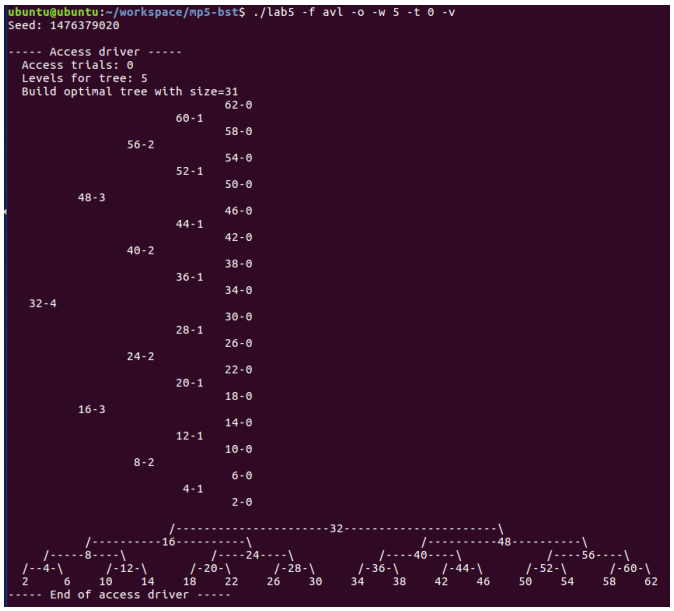
```
lab5 -o -w 5 -t 0 -v
// tests inserts only and prints tree
lab5 -r -w 5 -t 0 -v -s 1
// same with random tree
lab5 -p -w 5 -t 0 -v -s 2
// same with random tree
```

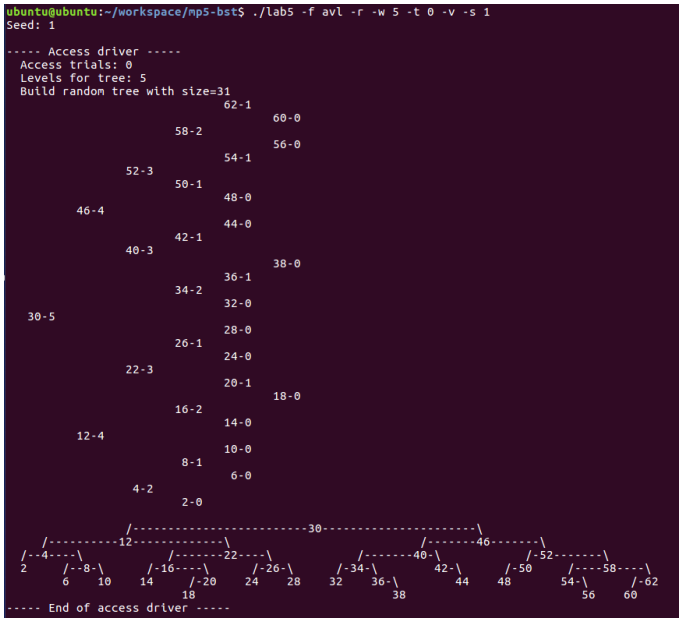
To validate that the tree remains in binary search tree's property: 1) bst

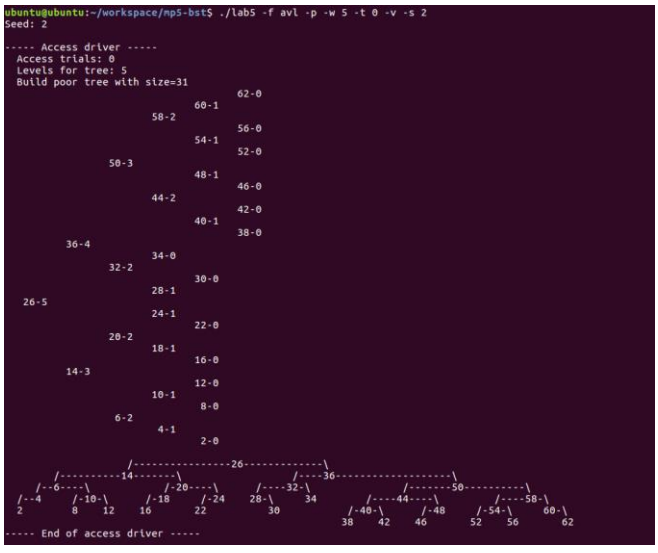




2) avl:


ubuntu@ubuntu:~/workspace/mp5-bst$ ./lab5 -f avl -o -w 5 -t 0 -v
Seed: 1476379020
----- Access driver -----
Access trials: 0
Levels for tree: 5
Build optimal tree with size=31
----- End of access driver -----

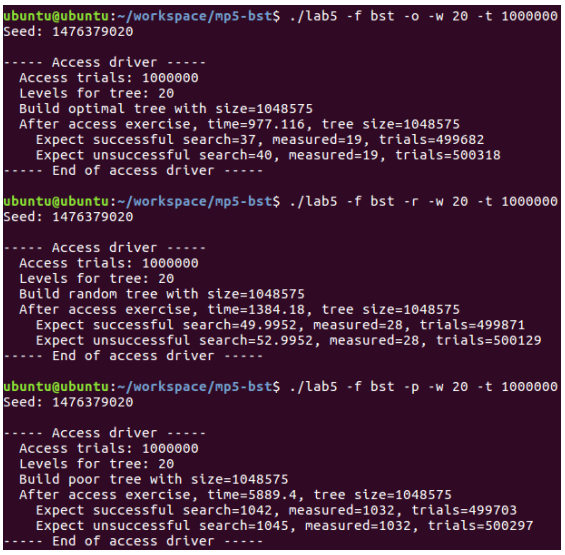

ubuntu@ubuntu:~/workspace/mp5-bst$ ./lab5 -f avl -r -w 5 -t 0 -v -s 1
Seed: 1
----- Access driver -----
Access trials: 0
Levels for tree: 5
Build random tree with size=31
----- End of access driver -----


ubuntu@ubuntu:~/workspace/mp5-bst$ ./lab5 -f avl -p -w 5 -t 0 -v -s 2
Seed: 2
----- Access driver -----
Access trials: 0
Levels for tree: 5
Build poor tree with size=31
----- End of access driver -----

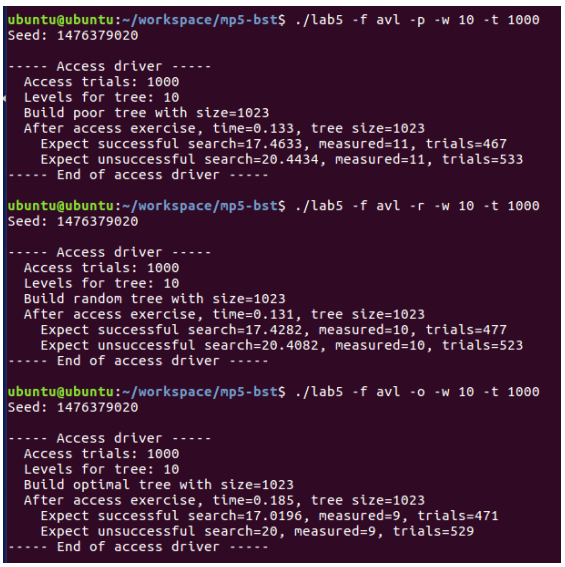
4. Do the command line arguments of:

```
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. 1) bst:



2) avl:



bst: best: 37, 40
average: 50, 53
worst: 1042, 1045

avl: best: 17, 20
average: 17, 20

worst: 17, 20

5. I find out that in standish's textbook:

Case	C_n	C_n'
Best	$2log_2 n - 3$	$log_2 n$
Average	$2.77log_2 n - 4.7$	$2.77log_2 n - 1.7$
Worst	n	n + 2

And what I found: (bst->20 levels)

Case	$C_{2^{20}}$ (Mine)	$C_{2^{20}}'$(Mine)
Best	$2log_2 2^{20} - 3$ = 37 (37)	$2log_2 2^{20}$ = 40 (40)
Average	$2.77log_2 2^{20} - 4.7$ = 50.7 (50)	$2.77log_2 2^{20} - 1.7$ = 53.7(53)
Worst	2^{20} (1000000) (1042)	2^{20} + 2 (1000002) (1045)

And what I found: (avl->10 levels)

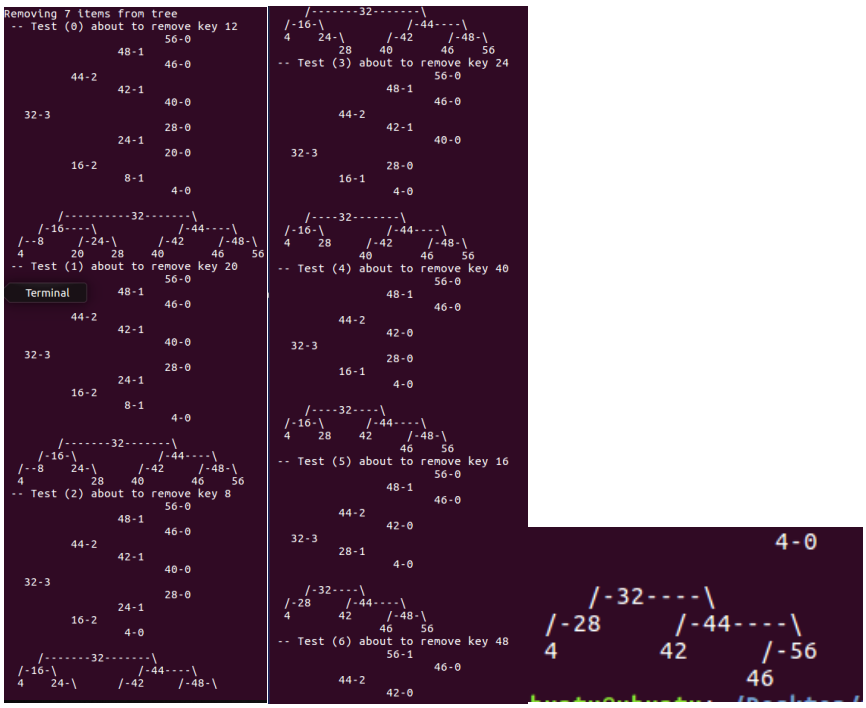
Case	$C_{2^{10}}$(Mine)	$C_{2^{10}}'$(Mine)
Best	$2log_2 2^{10} - 3$ = 17 (17)	$2log_2 2^{10}$ = 20 (20)
Average	$2.77log_2 2^{10} - 4.7$ = 23 (17)	$2.77log_2 2^{10} - 1.7$ = 26 (20)
Worst	2^{10} (1000) (17)	2^{20} + 2 (1002) (20)

1) So by comparation of my number of successful and unsuccessful searches, it just perfectly match the Approximate Average Number of Comparisons for successful and unsuccessful search , so that my binary search tree works well in three cases.
2) As for worst case, I guess its time of complexity is O($\sqrt{n}$), for it generates trees that only have two subtrees that looks like linked list. For the time complexity of remove, insert and access of a certain node in linked list is O(n), and the poor tree looks like ^, it cuts more than 2 times in every moves downward.
3) As for the avl tree because its balanced characteristic, so, no matter in what policy the node is inserted, the avl tree matches the best cases.

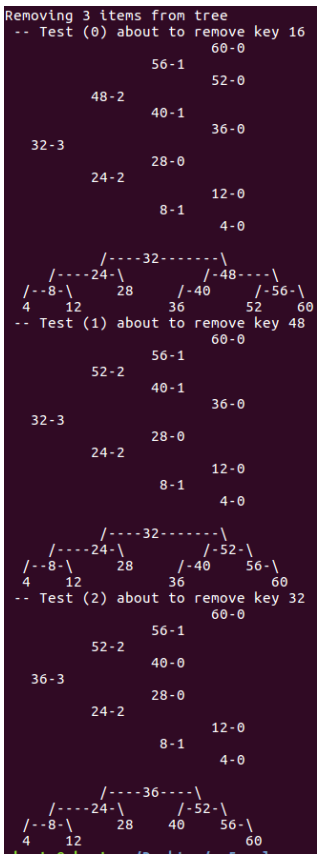
EXTENDED FOR REPLACEMENT MP!

1. Test through unit test driver as same as what I do for bst testing for No.1 test plan, but rather than do: ./lab5 -u 0-6, do: ./lab5 -f -avl -u 0-7

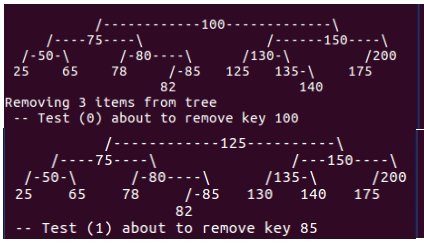
   (1)./lab5 -f -avl -u 0(first go up to down and then go right)

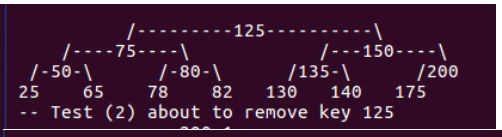


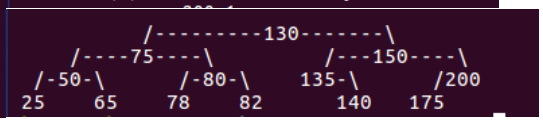
   (2)./lab5 -f -avl -u 1



(3)./lab5 -f avl -u 2

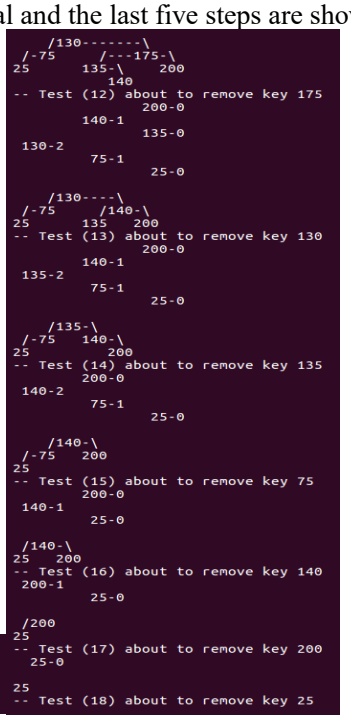


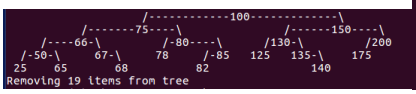




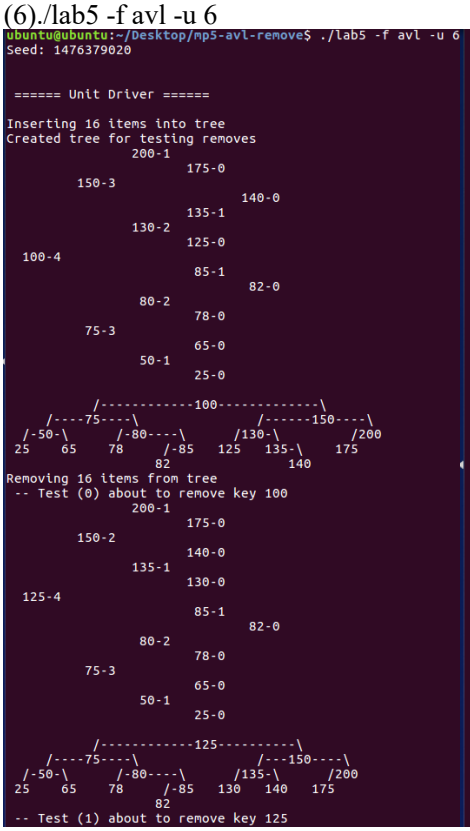
(4)./lab5 -f avl -u 3 and 4(just as same as bst, because only one node, do not involve with rotation)

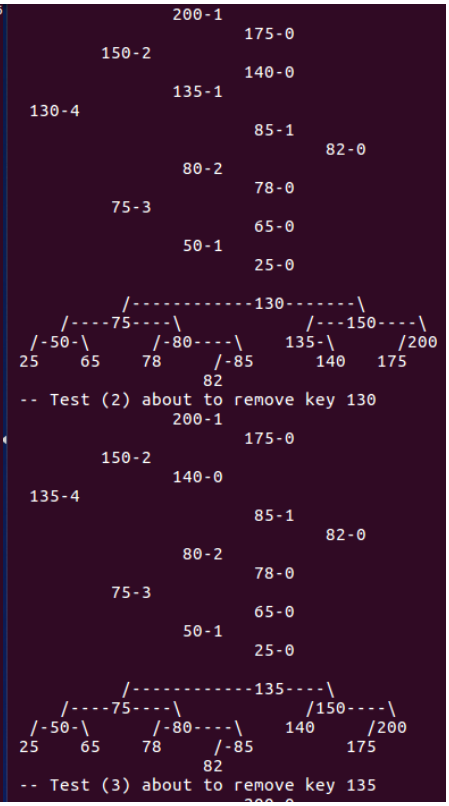
(5)./lab5 -f avl -u 5 (only the original and the last five steps are shown)

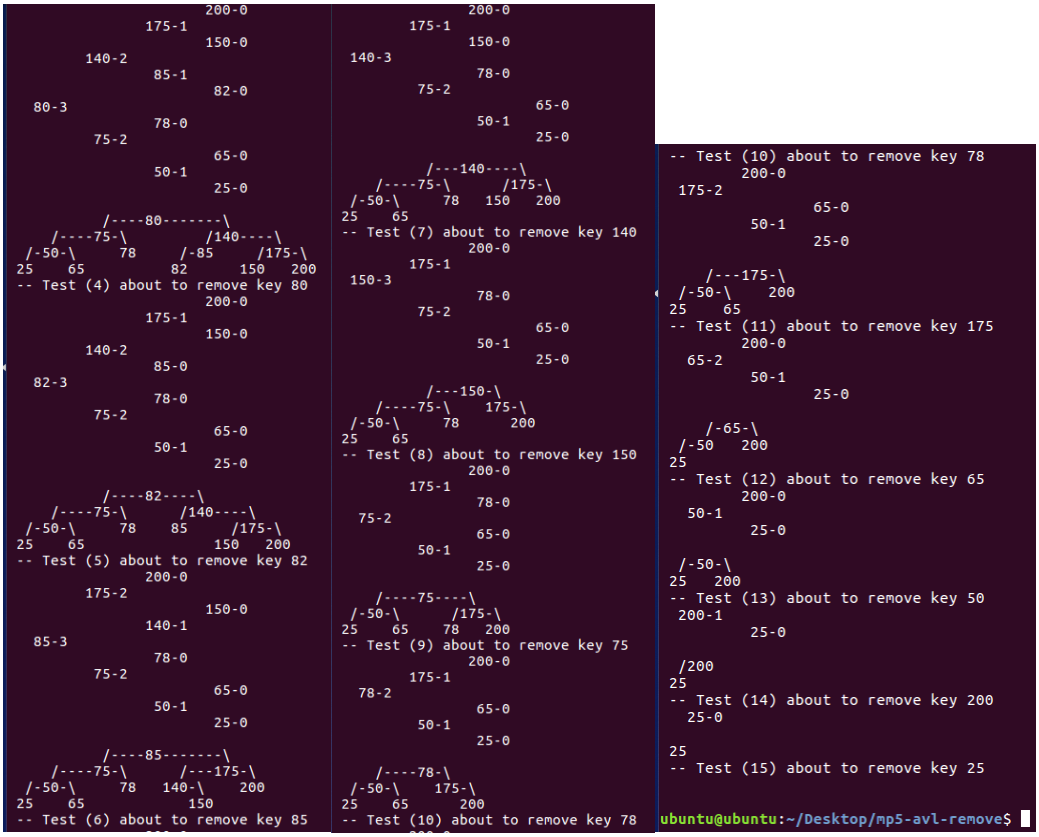




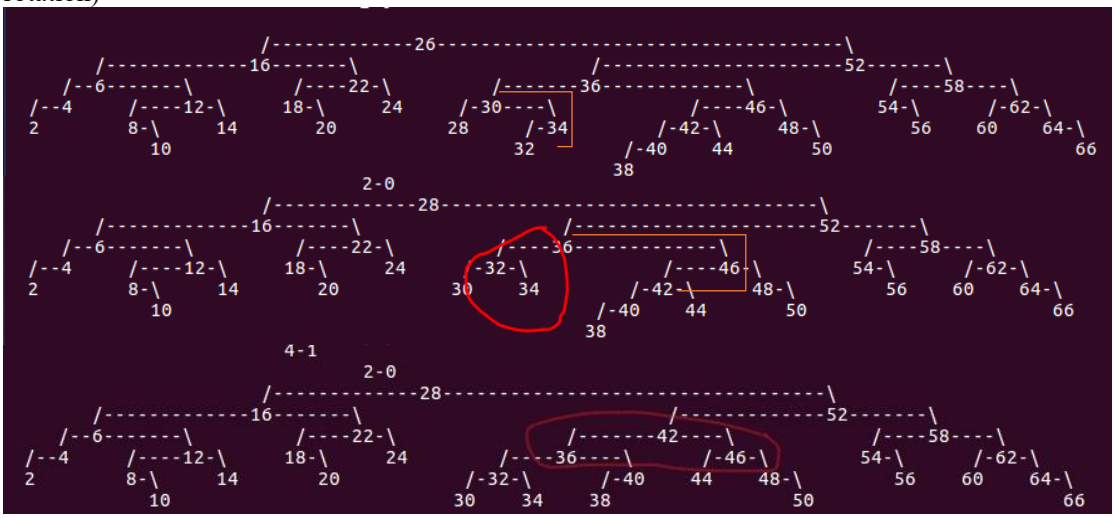
(6)./lab5 -f avl -u 6







(7) ./lab5 -f avl -u 7(Orignal, After 1st rotation, After 2rd rotations, every rotation is a double rotation)



2. For all the combination of options, since I have already done the BST deletion in test plan #4 and #5, so that I only do the command line of :

   "./lab5 -f avl -p -w 10 -t 1000"// performance evaluation(Like redo of #5)

   "./lab5 -f avl -r -w 10 -t 1000"

   "./lab5 -f avl -b -w 10 -t 1000"

   and example test:

   "./lab5 -f avl -e -w 16 -t 100000"// exercise tree (Like redo of #4)

   and

   "./lab5 -f avl -e -w 5 -t 10 -v -s 2"// tests random tree operations(Like redo of #3)

(1) ***"./lab5 -f avl -p -w 10 -t 1000"// performance evaluation(Like redo of #5)***

"./lab5 -f avl -r -w 10 -t 1000"

"./lab5 -f avl -b -w 10 -t 1000"

They are the same as what I have tested during performance evaluations.

```
ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -o -w 10 -t 1000
Seed: 1476379020

----- Access driver -----
  Access trials: 1000
  Levels for tree: 10
  Build optimal tree with size=1023
  After access exercise, time=0.129, tree size=1023
    Expect successful search=17.0196, measured=9, trials=471
    Expect unsuccessful search=20, measured=9, trials=529
----- End of access driver -----

ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -r -w 10 -t 1000
Seed: 1476379020

----- Access driver -----
  Access trials: 1000
  Levels for tree: 10
  Build random tree with size=1023
  After access exercise, time=0.128, tree size=1023
    Expect successful search=17.4282, measured=10, trials=477
    Expect unsuccessful search=20.4082, measured=10, trials=523
----- End of access driver -----

ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -p -w 10 -t 1000
Seed: 1476379020

----- Access driver -----
  Access trials: 1000
  Levels for tree: 10
  Build poor tree with size=1023
  After access exercise, time=0.126, tree size=1023
    Expect successful search=17.4633, measured=11, trials=467
    Expect unsuccessful search=20.4434, measured=11, trials=533
----- End of access driver -----

ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -e -w 10 -t 1000
Seed: 1476379020

----- Equilibrium test driver -----
  Trials in equilibrium: 1000
  Levels in initial tree: 10
  Initial random tree size=1023
  Expect successful search for initial tree=17.4282
  Expect unsuccessful search for initial tree=20.4082
  After exercise, time=13.847, new tree size=1023
  successful searches during exercise=9.1996, trials=506
  unsuccessful searches during exercise=9.80769, trials=494
  Validating tree...passed
  After access experiment, time=0.133, tree size=1023
  Expect successful search=17.4418, measured=9, trials=501
  Expect unsuccessful search=20.4219, measured=9, trials=499
----- End of equilibrium test -----
```

(2) ***"./lab5 -f avl -e -w 16 -t 100000"// exercise tree (Like redo of #4)***

That is the ultimate limit short running time of my avl tree, I cannot wait further (eg. I really don't know how long will it take beyond that). If you run 20 levels with 1,000,000 trials, you might wait for hours.

```
ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -e -w 16 -t 100000
Seed: 1476379020

----- Equilibrium test driver -----
  Trials in equilibrium: 100000
  Levels in initial tree: 16
  Initial random tree size=65535
  Expect successful search for initial tree=29.7222
  Expect unsuccessful search for initial tree=32.7217
  After exercise, time=248699, new tree size=65699
  successful searches during exercise=15.356, trials=50078
  unsuccessful searches during exercise=15.9348, trials=49922
  Validating tree...passed
  After access experiment, time=34.02, tree size=65699
  Expect successful search=29.6901, measured=17, trials=50285
  Expect unsuccessful search=32.6896, measured=17, trials=49715
----- End of equilibrium test -----
```

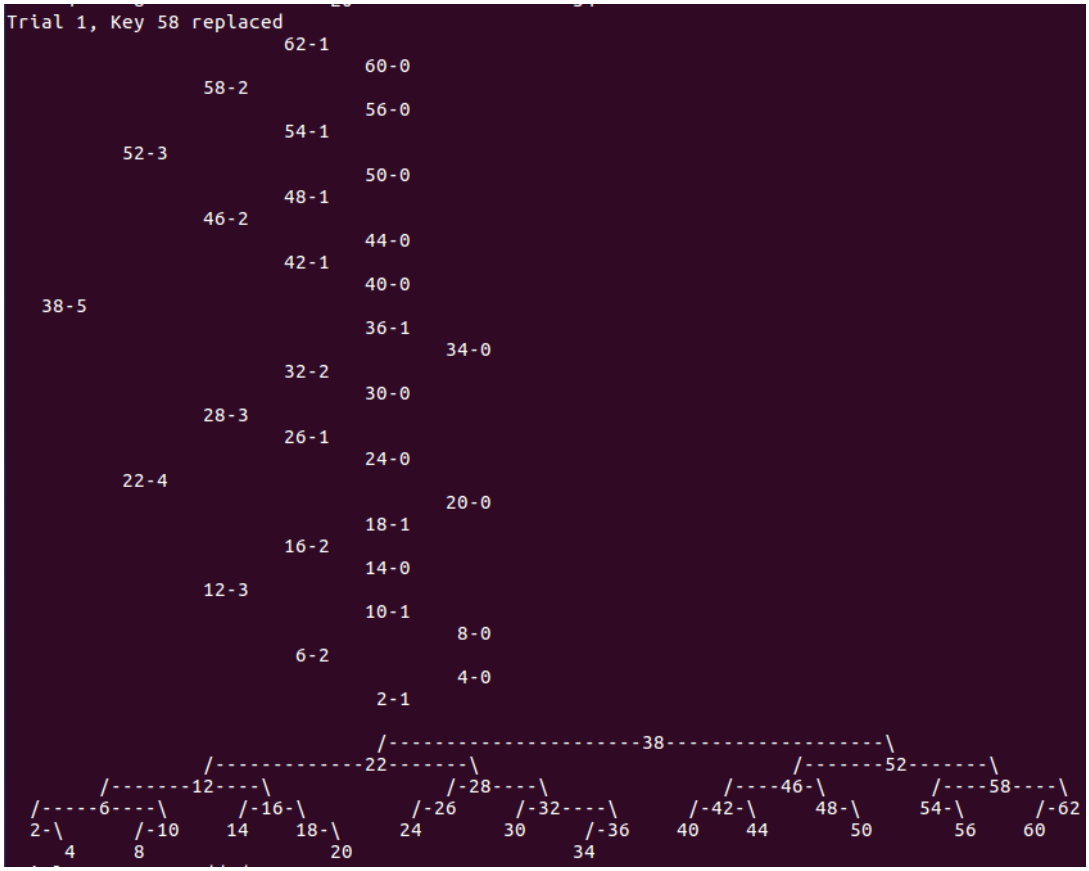
(3) ***"./lab5 -f avl -e -w 5 -t 10 -v -s 2"// tests random tree operations(Like redo of #3)***

```
ubuntu@ubuntu:~/Desktop/mp5-avl-remove$ ./lab5 -f avl -e -w 5 -t 10 -v -s 2
Seed: 2

----- Equilibrium test driver -----
  Trials in equilibrium: 10
  Levels in initial tree: 5
  Initial random tree size=31
  Expect successful search for initial tree=7.58065
  Expect unsuccessful search for initial tree=10.25
                        62-1
                                60-0
                58-2
                                56-0
                        54-1
        52-3
                                50-0
                        48-1
                46-2
                                44-0
                        42-1
                                40-0
   38-5
                                36-1
                                        34-0
                        32-2
                                30-0
                28-3
                        26-1
                                24-0
        22-4
                                        20-0
                                18-1
                        16-2
                                14-0
                12-3
                                10-1
                                         8-0
                         6-2
                                         4-0
                                 2-1

                                /----------------------38------------------\
                /-------------22-------\                                /-------52-------\
       /--------12----\                /-28----\                    /----46-\         /----58----\
  /-----6----\      /-16-\          /-26     /-32----\      /-42-\     48-\     54-\       /-62
  2-\      /-10    14    18-\     24        30      /-36    40    44        50       56    60
     4     8                20                     34
```

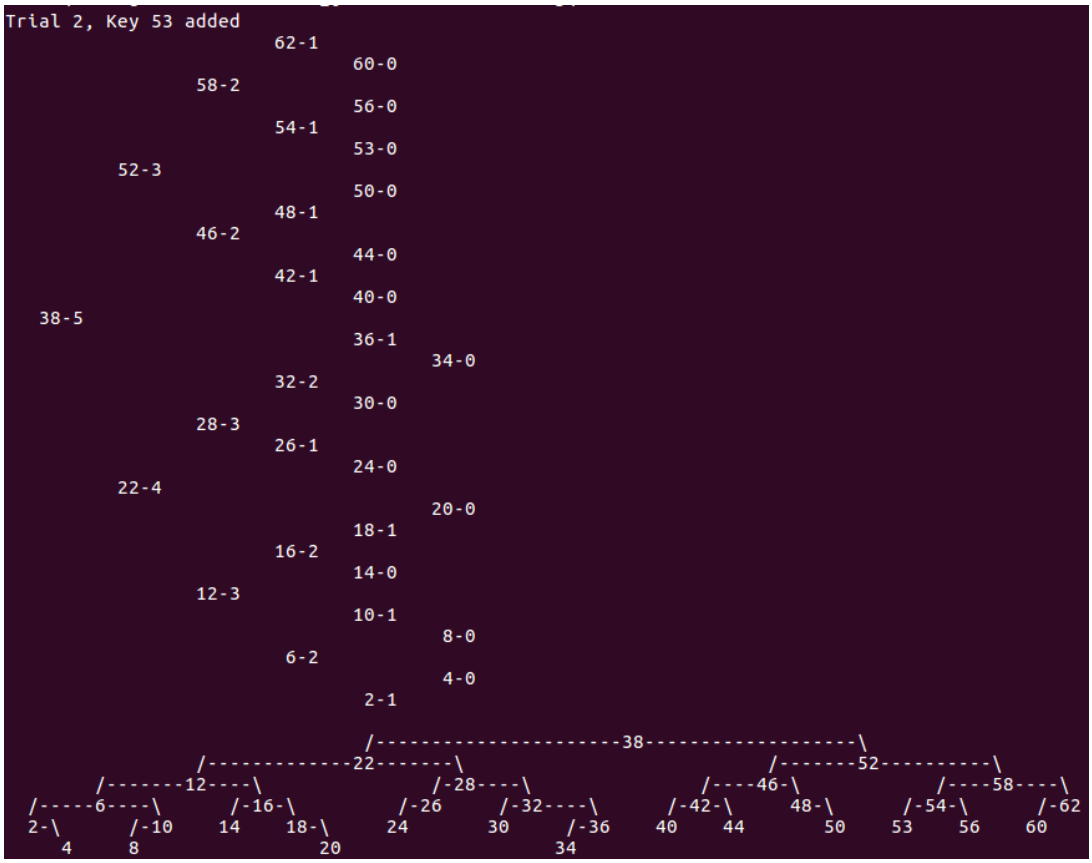
Trial 0, key 2 replaced

```
Trial 0, Key 2 replaced
                        62-1
                                60-0
                58-2
                                56-0
                        54-1
        52-3
                                50-0
                        48-1
                46-2
                                44-0
                        42-1
                                40-0
   38-5
                                36-1
                                        34-0
                        32-2
                                30-0
                28-3
                        26-1
                                24-0
        22-4
                                        20-0
                                18-1
                        16-2
                                14-0
                12-3
                                10-1
                                         8-0
                         6-2
                                         4-0
                                 2-1

                                /----------------------38------------------\
                /-------------22-------\                                /-------52-------\
       /--------12----\                /-28----\                    /----46-\         /----58----\
  /-----6----\      /-16-\          /-26     /-32----\      /-42-\     48-\     54-\       /-62
  2-\      /-10    14    18-\     24        30      /-36    40    44        50       56    60
     4     8                20                     34
```

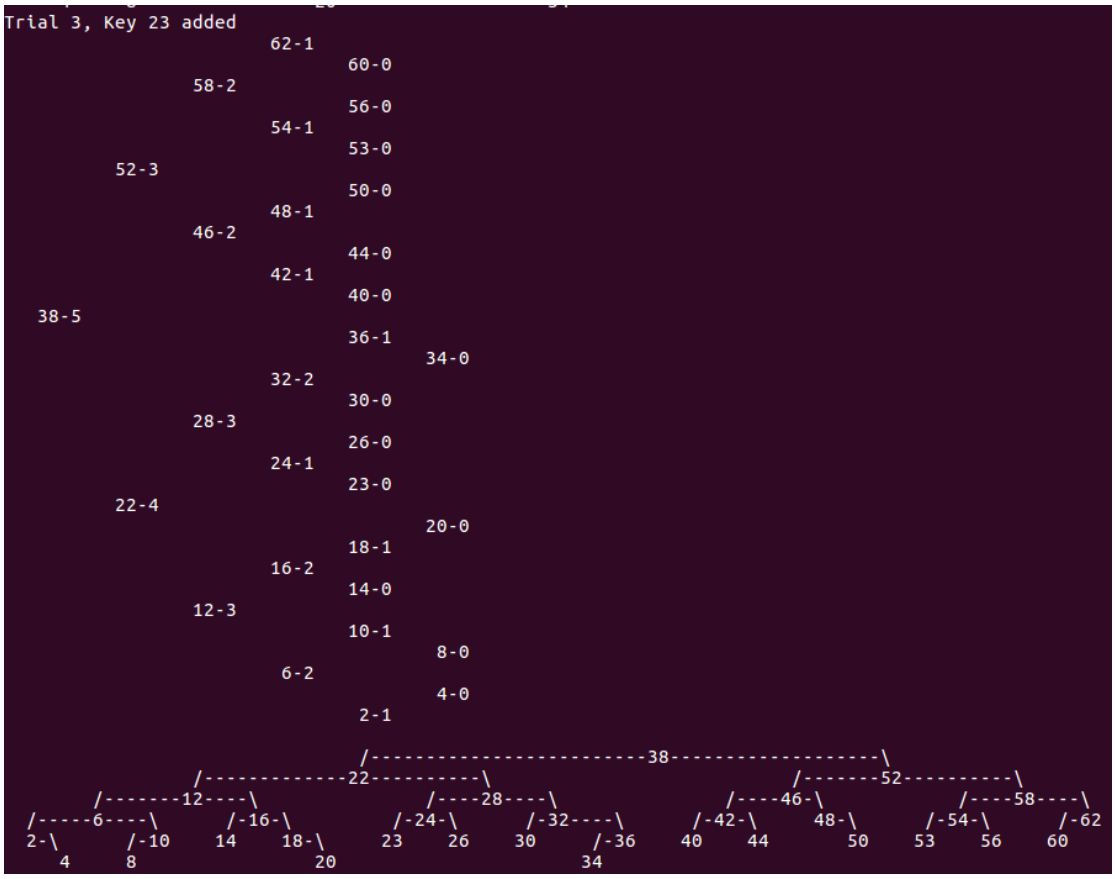
Trial 1, key 58 replaced



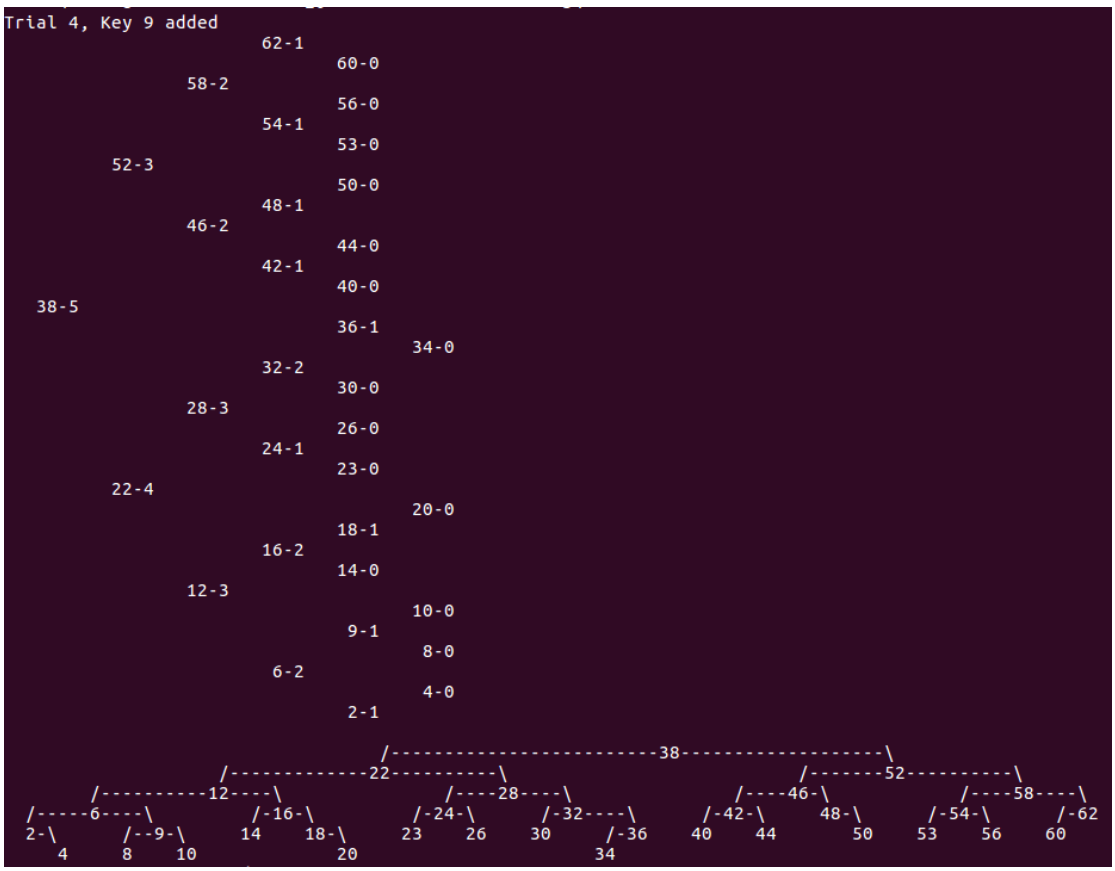
Trial 2, key 53 added



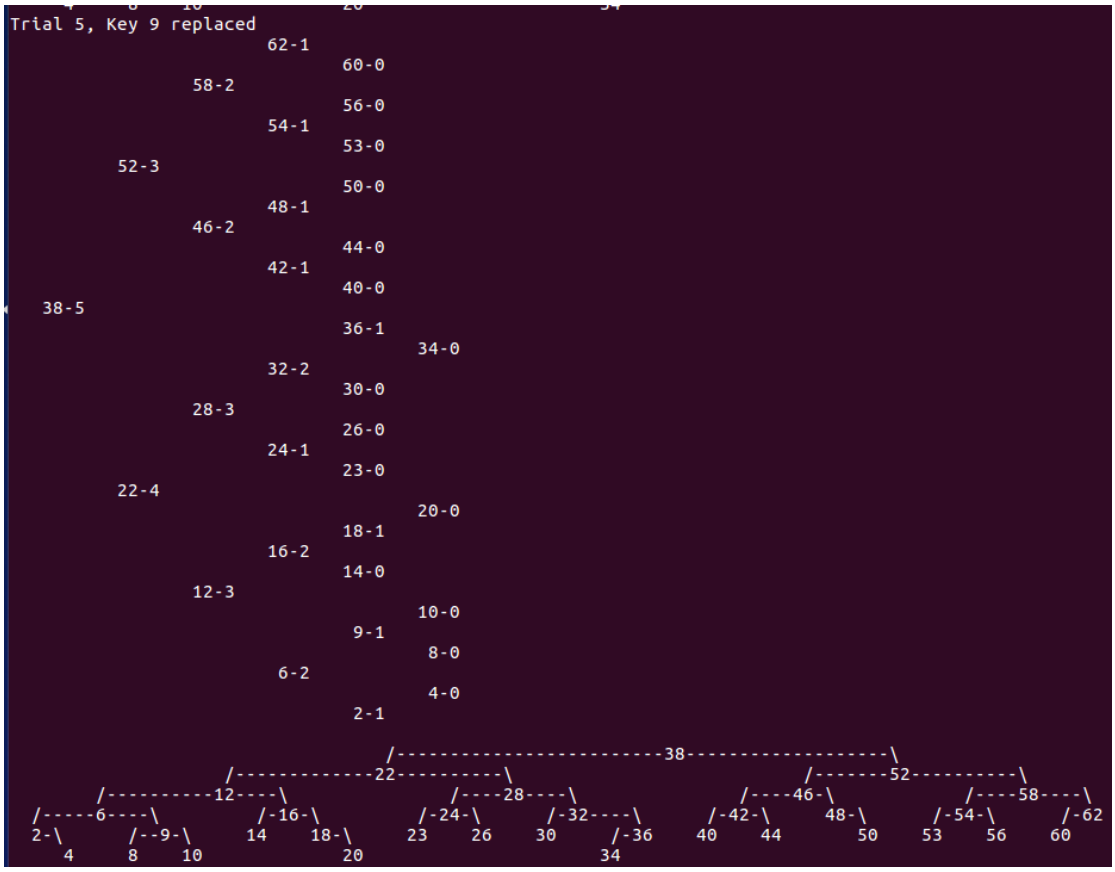
Trial 3, key 23 added



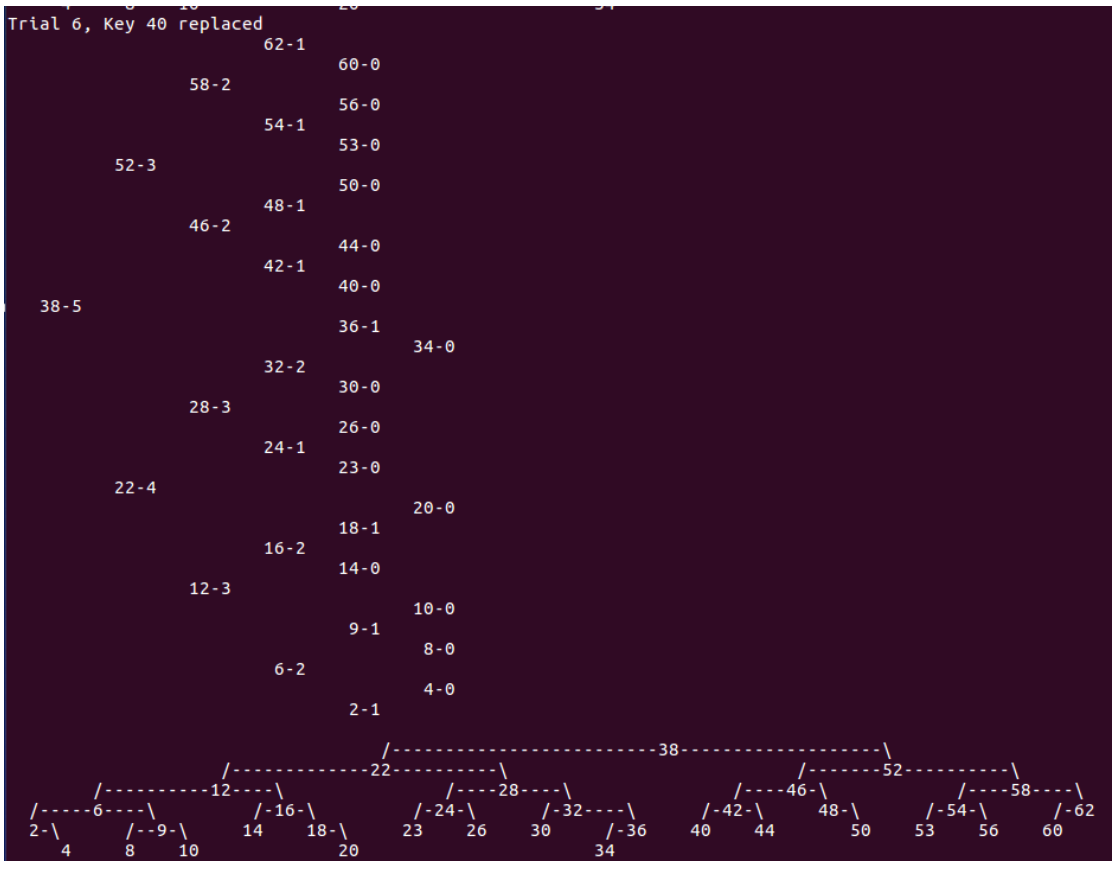
Trial 4, key 9 added



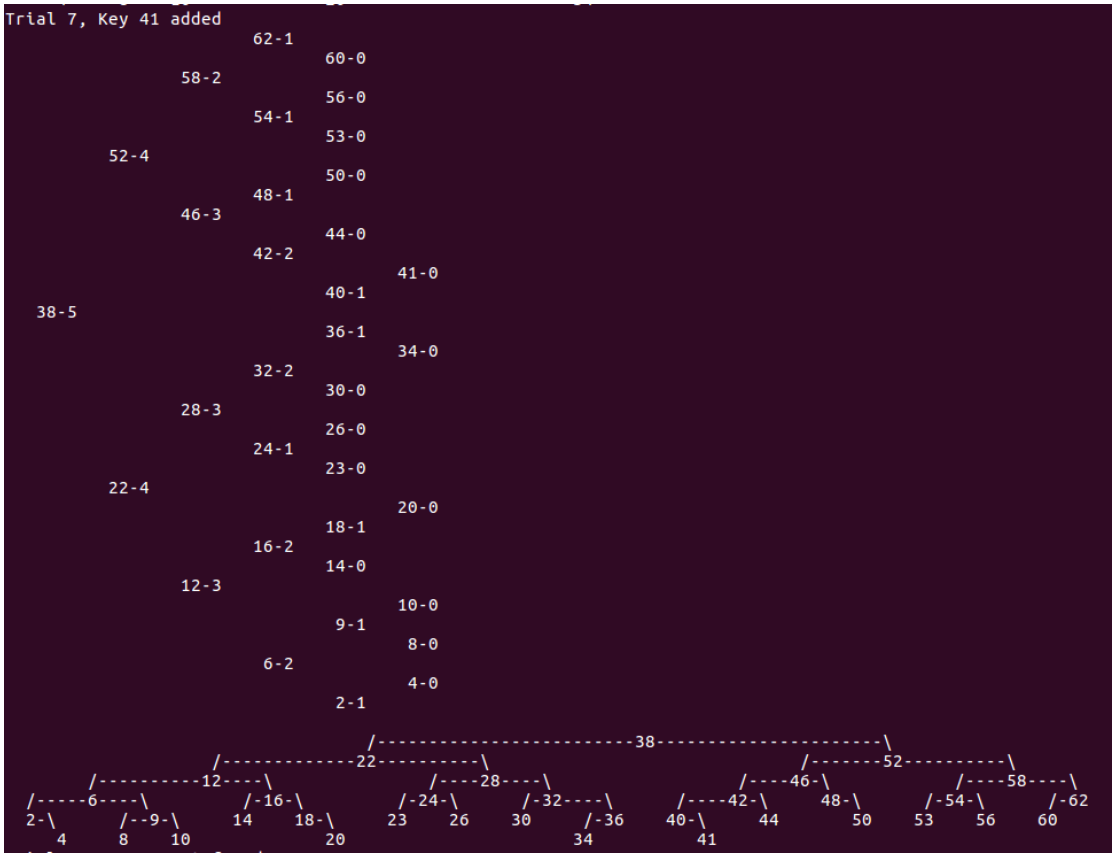
Trial 5, key 9 replaced



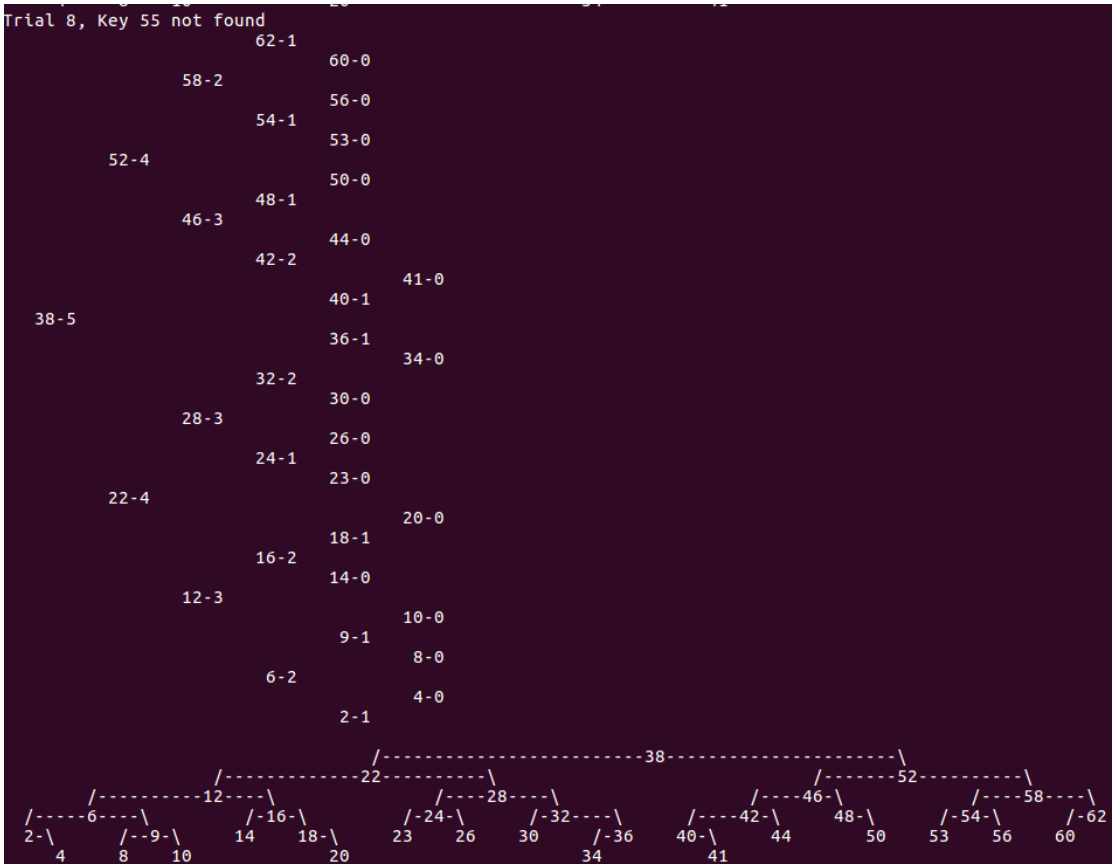
Trial 6, key 40 replaced



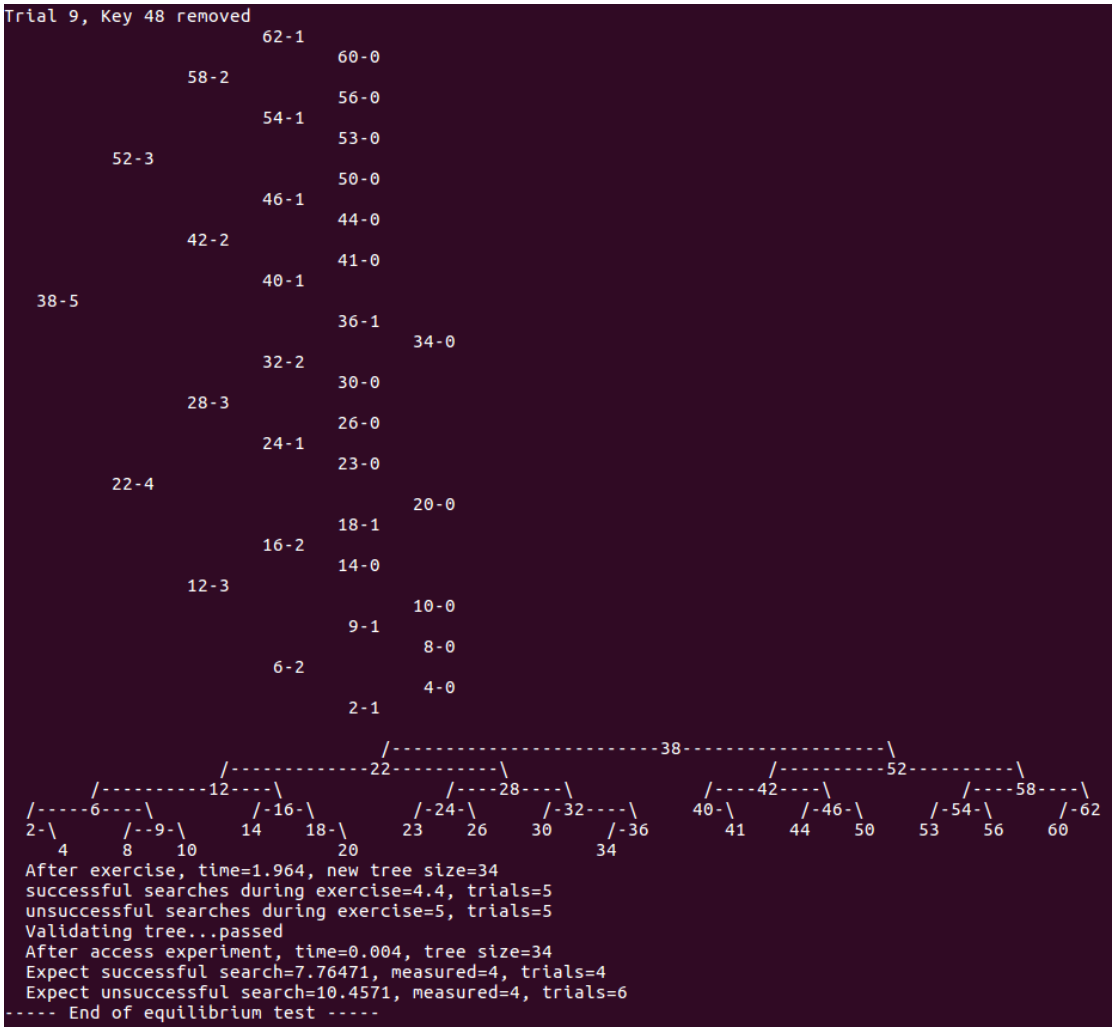
Trial 7, key 41 added



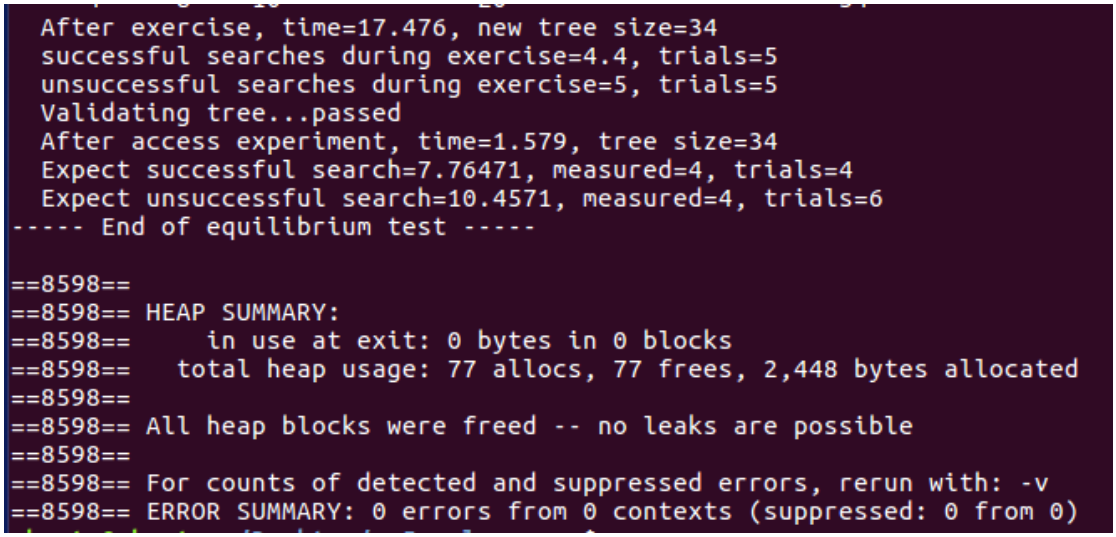
Trial 8, key 55 not found



Trial 9, Key 48 removed



Bonus: leak test!



/*<----------THE END OF TEST LOG FOR MP5 AND REPLACEMENT MP---------->*/