Elif Gülşah Kaşdoğan/21601183

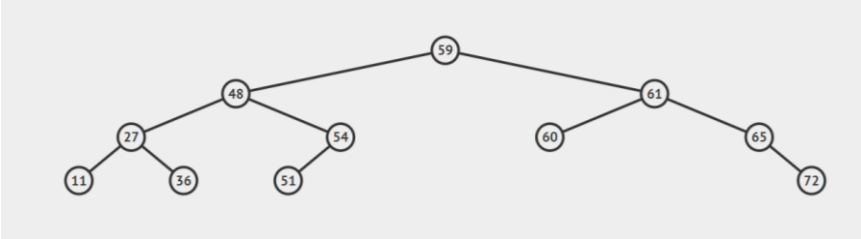
1.

A. Inorder: 3 – 5*8 / 4 ^ 1 + 7

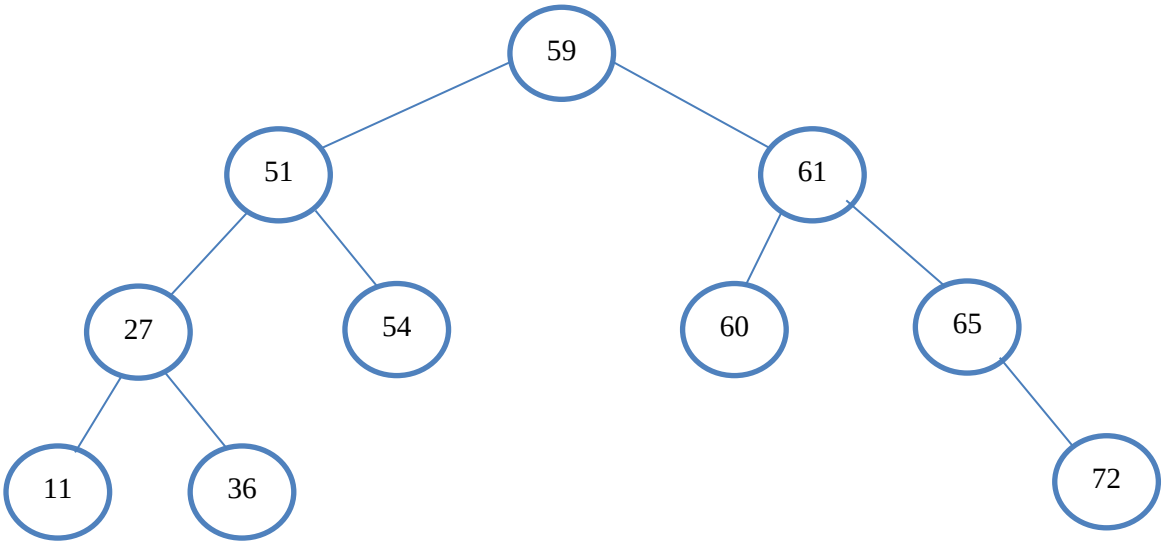
Preoder: / - 3 * 5 8 + ^ 4 1 7

Postorder: 3 5 8 * - 4 1 ^ 7 + /

B. VisualAlgo result

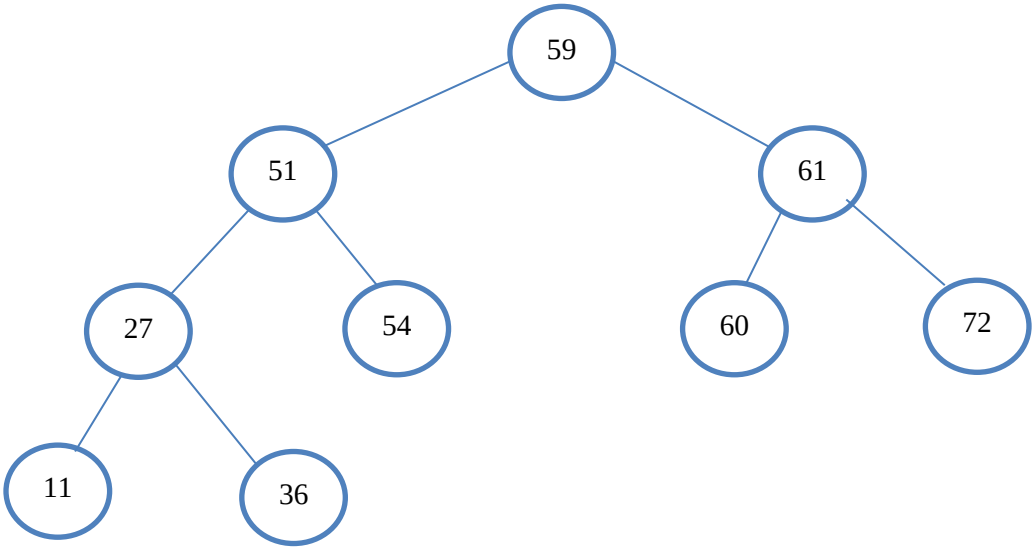


Delete 48

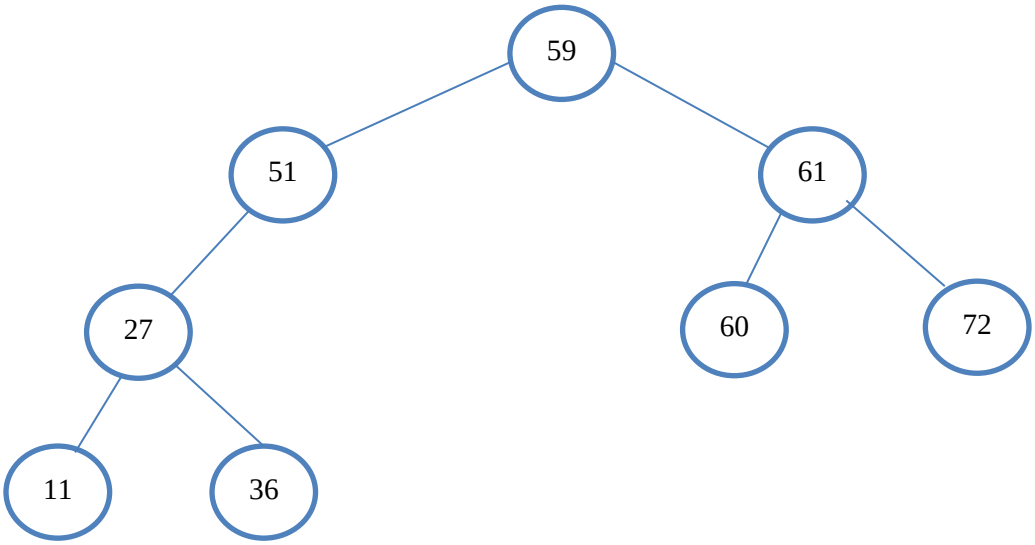


Elif Gülşah Kaşdoğan/21601183

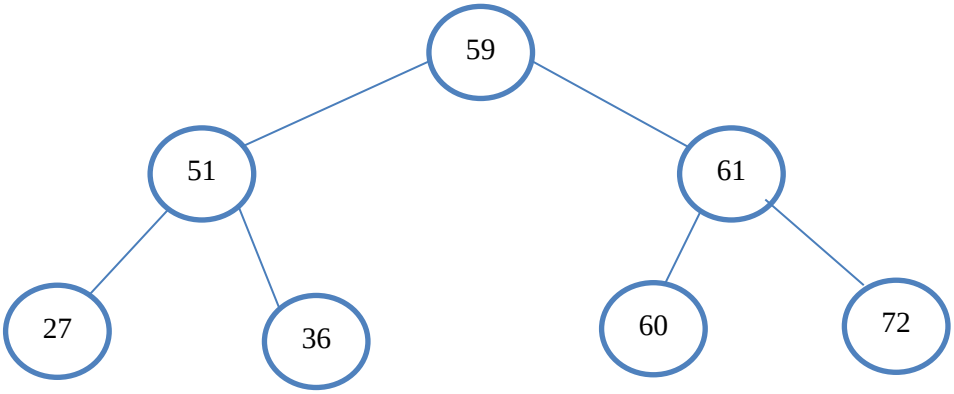
Delete 65



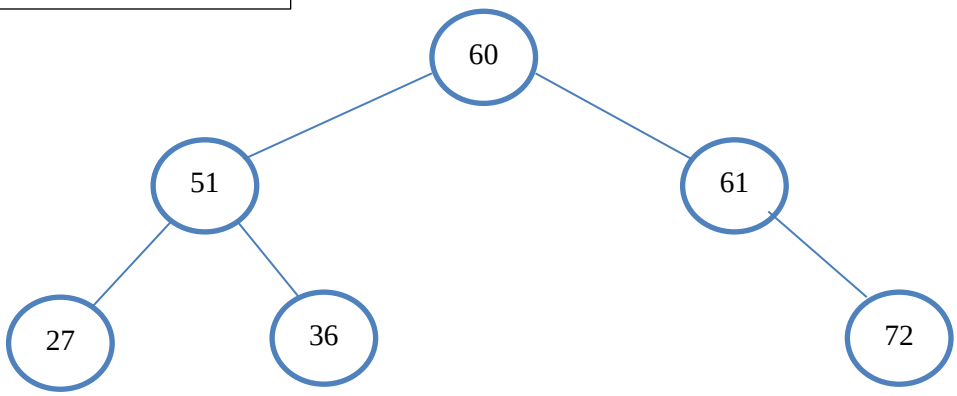
Delete 54



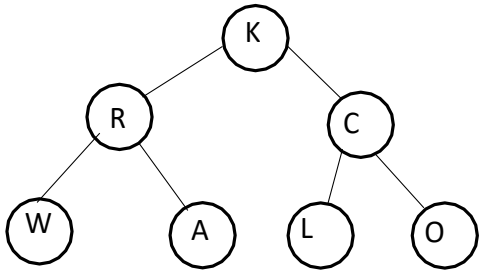
Delete 11



Delete 59



C. Postorder: W;A;R; L;O;C;K.

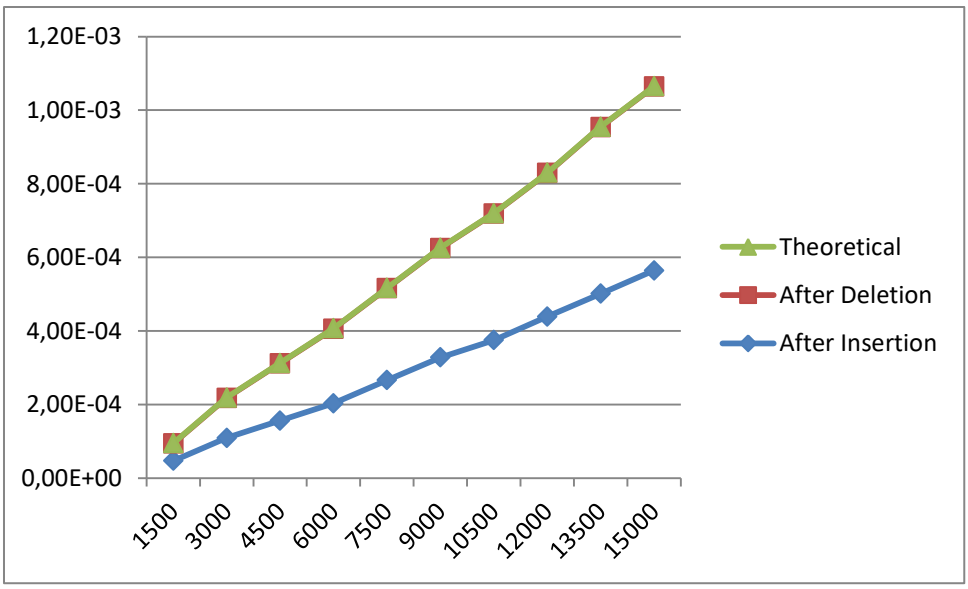


Inorder: W R A K L C O

3.

```
Part e - Time analysis of Binary Search Tree - part 1
-------------------------------------------------------
Tree Size         Time Elapsed
-------------------------------------------------------
1500          4.7e-005
3000          0.000109
4500          0.000156
6000          0.000203
7500          0.000266
9000          0.000328
10500          0.000375
12000          0.000439
13500          0.000501
15000          0.000564
Part e - Time analysis of Binary Search Tree - part 2
-------------------------------------------------------
Tree Size         Time Elapsed
-------------------------------------------------------
1500          4.7e-005
3000          0.000109
4500          0.000156
6000          0.000203
7500          0.00025
9000          0.000297
10500          0.000344
12000          0.00039
13500          0.000453
15000          0.0005

Process returned 0 (0x0)   execution time : 1.267 s
Press any key to continue.
```



Elif Gülşah Kaşdoğan/21601183

Analysis Report:

Binary search tree insertion and deletion methods work in O(n) time in theory. In average, algorithms for deletion and insertion works in O(h) time where h is height. There was not that much difference between theoretical results and deletion operation I have performed but there were differences between insertion and theoretical results. I have checked the elapsed time after every 1500 insertions and marked these time results in table.

	After Inser	After Dele	Theoretical	Height
1500	4,70E-05	4,70E-05	7,83E-07	25
3000	0,000109	0,000109	8,77E-07	28
4500	0,000156	0,000156	9,09E-07	29
6000	0,000203	0,000203	9,71E-07	31
7500	0,000266	0,00025	9,71E-07	31
9000	0,000328	0,000297	9,71E-07	31
10500	0,000375	0,000344	9,71E-07	31
12000	0,000439	0,00039	1,00266E-06	32
13500	0,000501	0,000453	1,03399E-06	33
15000	0,000564	0,0005	1,03399E-06	33

I outputted the height of the binary search tree after every insertion operation and calculated the theoretical time with respect to that in milliseconds, I used 4,70E-05/1500 as unit ie. I obtained the theoretical results by multiplying height with that constant.

I followed the methodology we have learnt in class, the reason why deletion and theoretical results are similar might be related to shuffling, I created the entries of tree randomly, I shuffled the numbers in a random manner. Probably input entries were nicely uniformly distributed because if they were not I would see that the insertion results would be close to the theoretical one as deletion. Reason might be the following, shuffling created a structure similar to sorted entries case for this run, that is why deletion line became closer to the theoretical results whereas insertion did not. Of course we cannot conclude that this is the main reason without looking at the numbers generated, so it is just a theory.

If we use sorted numbers instead of unsorted ones, Binary Search Tree functions' will perform in their worst case because whether ascending or descending order sorted input will lead binary search tree to have max height, as I mentioned above since algorithm works in O(h) time it will hit the worst case.