Lab 2 Report

Name: Basel Ahmed Awad ID: 19015513

Name: Ali Hassan Ali Ahmed ELSharawy ID: 19016013

We implement node class with attributes value, parent, left child, right child and boolean attribute is Red to indicate red color at true with getters and setters for these attributes.

Also Nil class extends Node object as special node.

For Red Black Tree we have tree size, root, object of rotation handler which is responsible for rotation and inserted node to help at insertion function.

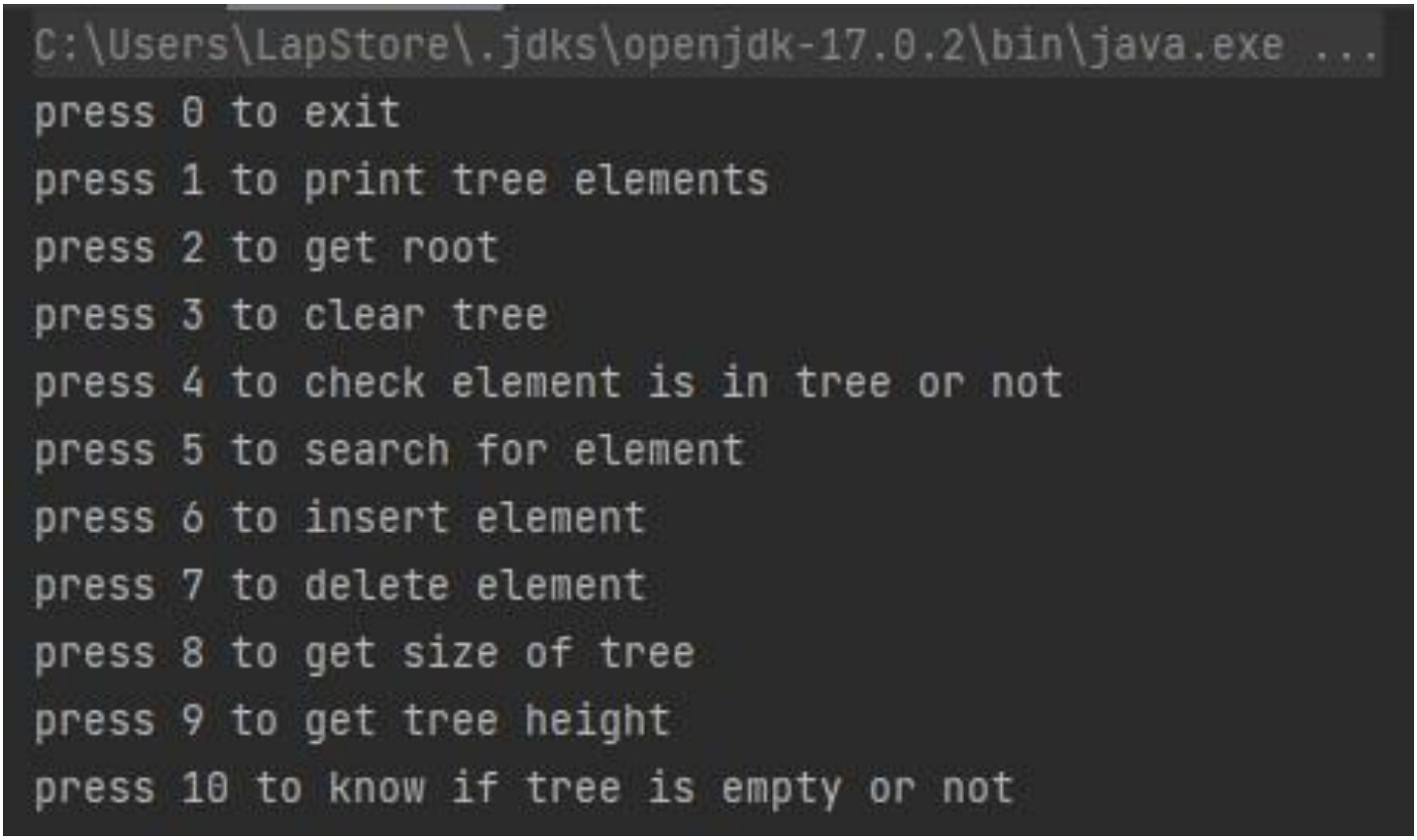
We have functions to print elements, get root, get height of tree, get size of tree, is Empty, contains that check if element is found or not.

Also search function same as AVL Tree and insertion that use validate function to handle tree according to parent’s and parent’s sibling color.

For deletion we have function to delete node with zero or one child also 2 children case use it after replacing node with its inorder successor.

For comparing time between Red Black Tree and AVL Tree we use function string generator which generate random string among small and capital alphabet and numbers from 0 to 10 and we generate n words to insert and delete to compare between them.

Test Cases:


C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
press 0 to exit
press 1 to print tree elements
press 2 to get root
press 3 to clear tree
press 4 to check element is in tree or not
press 5 to search for element
press 6 to insert element
press 7 to delete element
press 8 to get size of tree
press 9 to get tree height
press 10 to know if tree is empty or not

```
2
tree is empty
10
true
8
0
9
0
7
ssd
false
5
ali
not found
4
ahmed
false
0
```

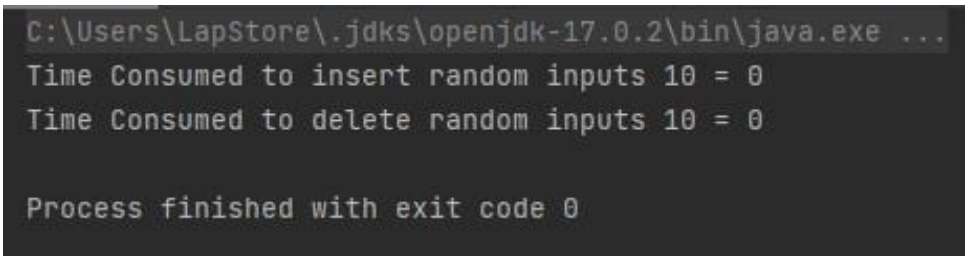
```
6
ali
true
6
ahmed
true
6
basel
true
6
ali
false
4
ali
true
5
basel
basel
8
3
2
ali
```

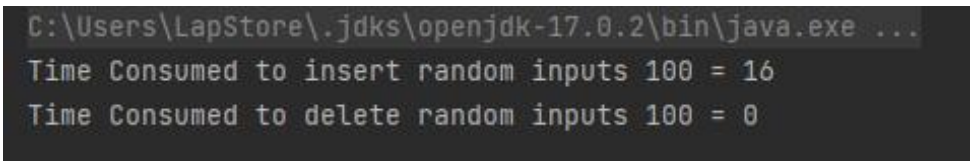
```
6
tea
true
6
cup
true
6
ball
true
2
cup
1
ball Red Node
cup Black Node
tea Red Node
4
tae
false
4
tea
true
5
tea
tea
```

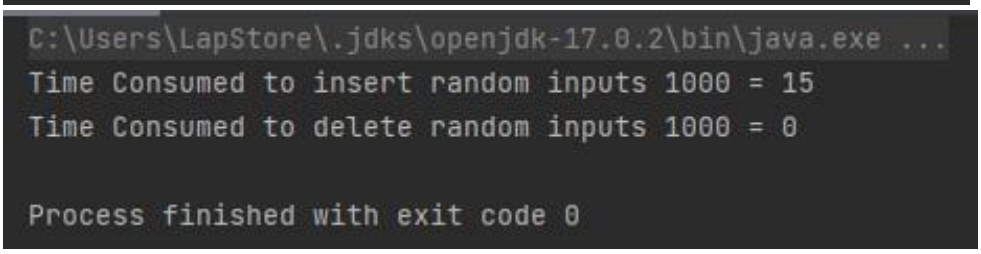
```
5
toe
not found
8
3
10
false
7
tea
true
7
cup
true
2
ball
1
ball Black Node
5
ball
ball
5
tea
not found
```

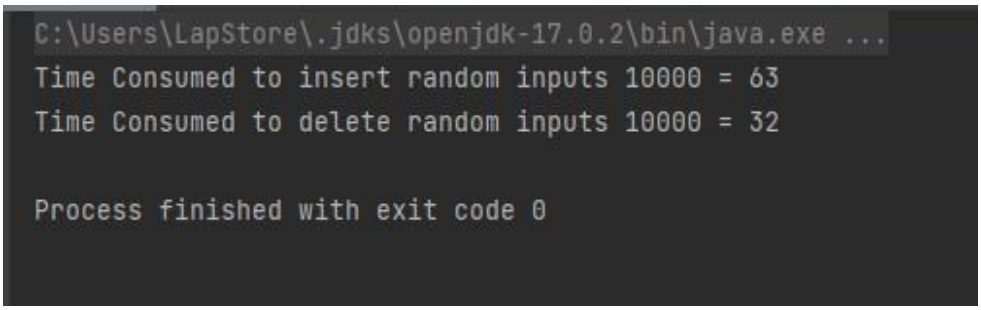
AVL Tree & Red Black Tree:

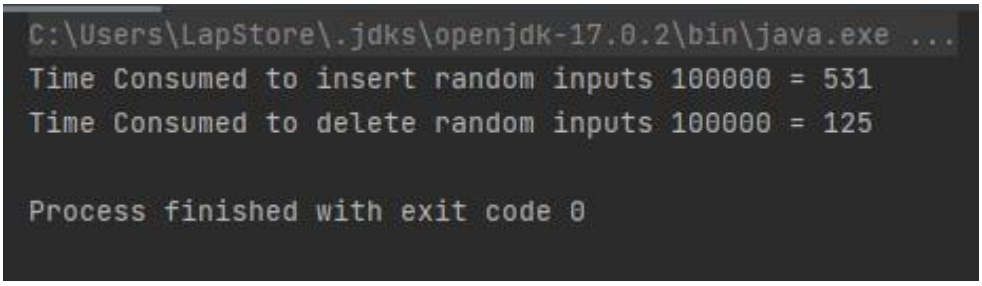
For Red Black Tree:











For AVL Tree:

```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 10 random input= 0
Time Consumed for deletion 10 random input = 0

Process finished with exit code 0
```

```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 100 random input= 16
Time Consumed for deletion 100 random input = 0

Process finished with exit code 0
```

```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 1000 random input= 15
Time Consumed for deletion 1000 random input = 0

Process finished with exit code 0
```

```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 10000 random input= 78
Time Consumed for deletion 10000 random input = 63

Process finished with exit code 0
```

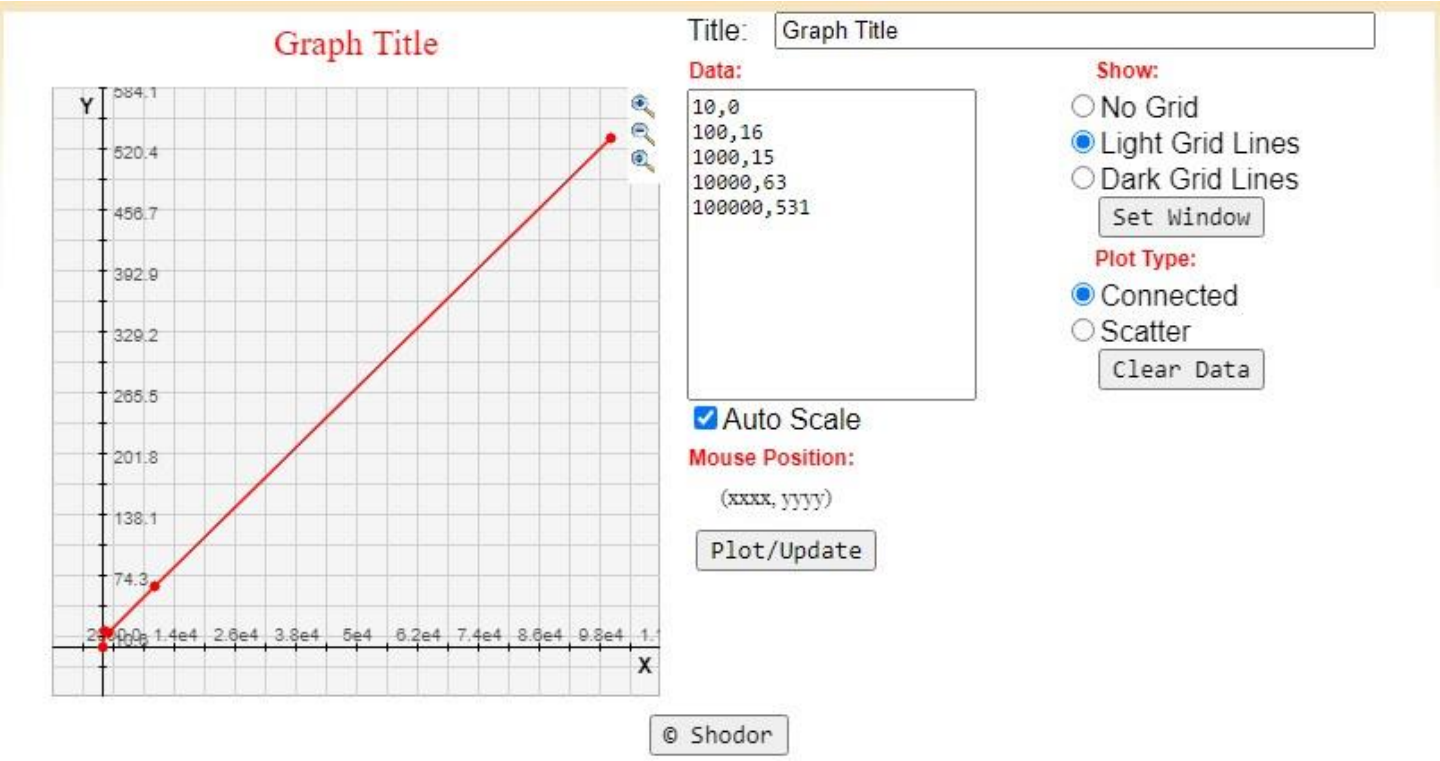
```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 100000 random input= 812
Time Consumed for deletion 100000 random input = 578

Process finished with exit code 0
```

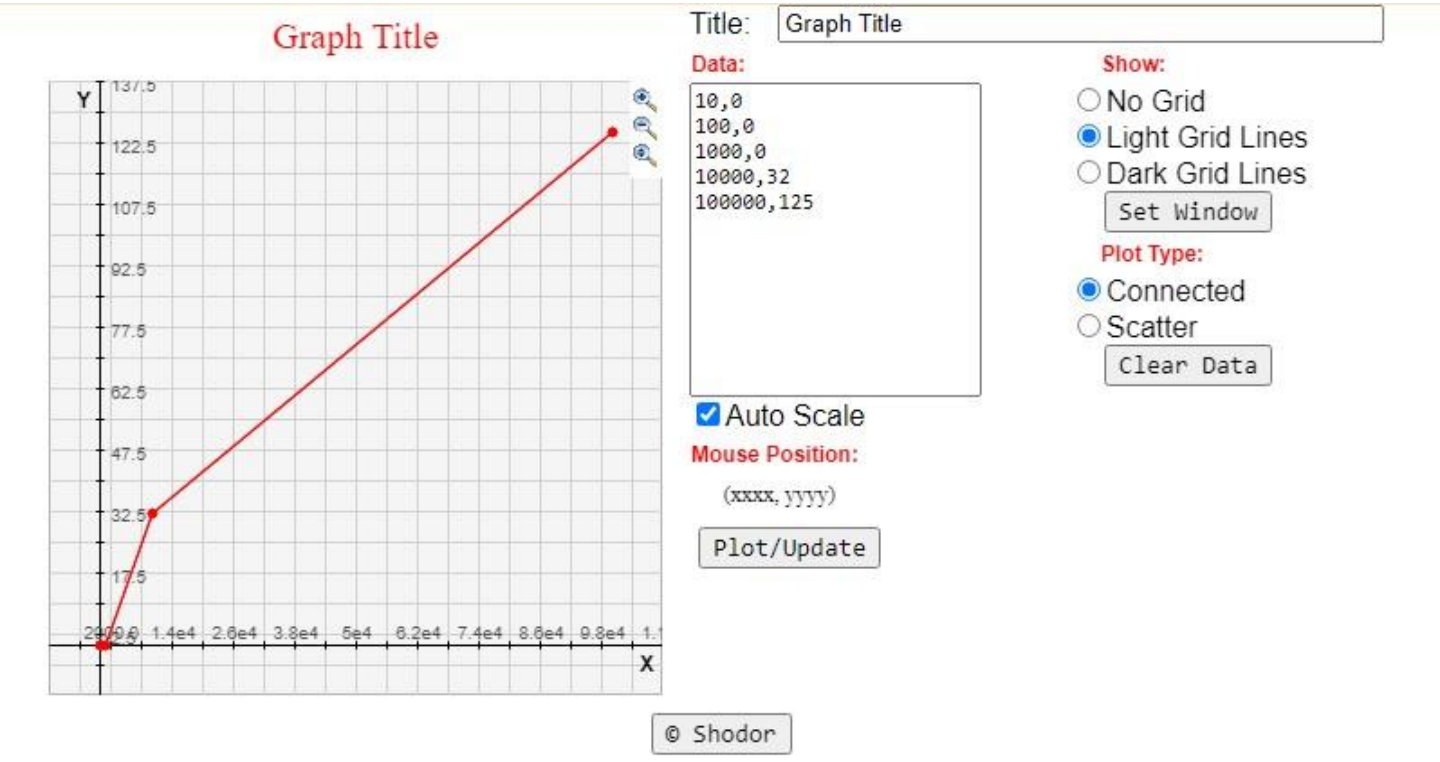
```
C:\Users\LapStore\.jdks\openjdk-17.0.2\bin\java.exe ...
Time Consumed for insertion 100000 random input= 594
Time Consumed for deletion 100000 random input = 297

Process finished with exit code 0
```

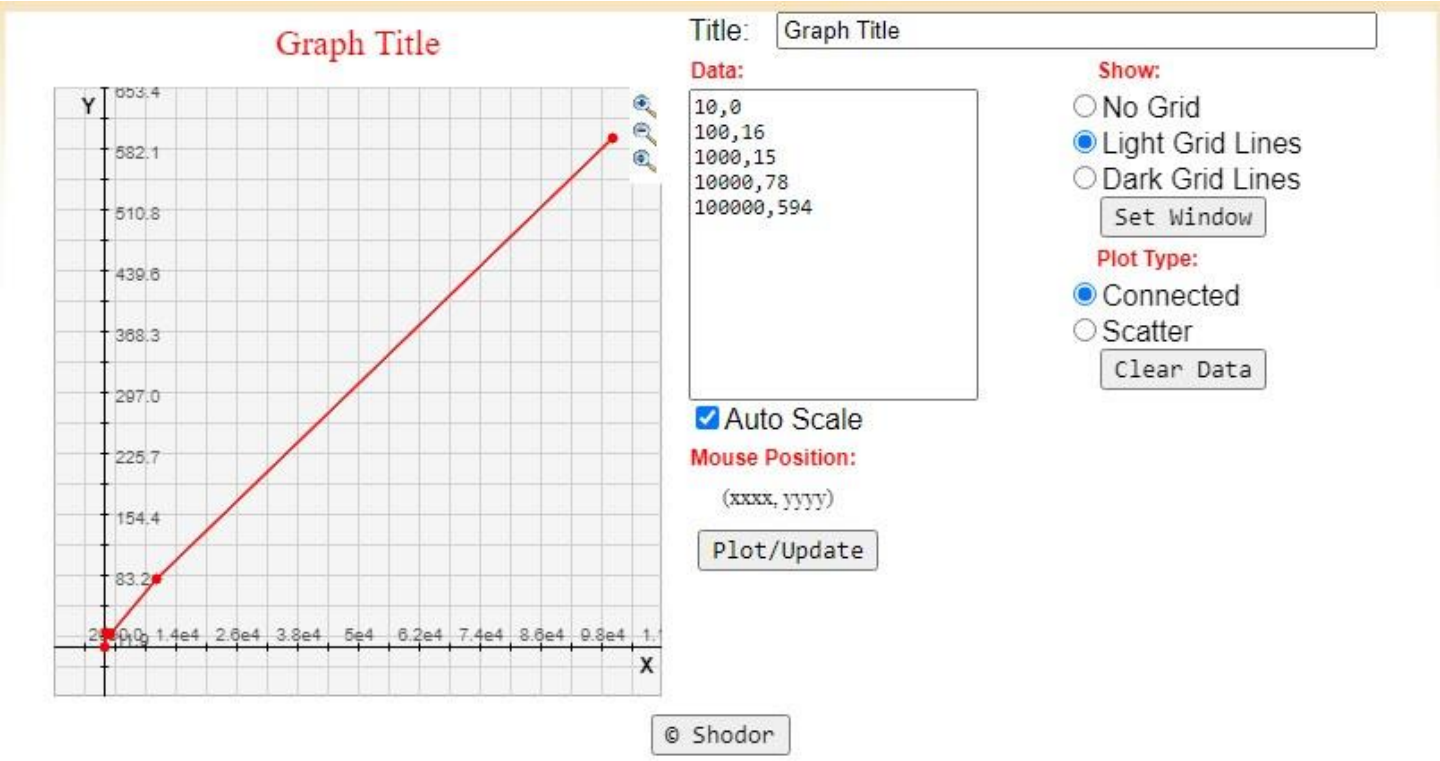
Red Black tree insertion graph:



Red Black tree deletion graph:



AVL tree insertion graph:



AVL tree deletion graph: