GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 7 Report

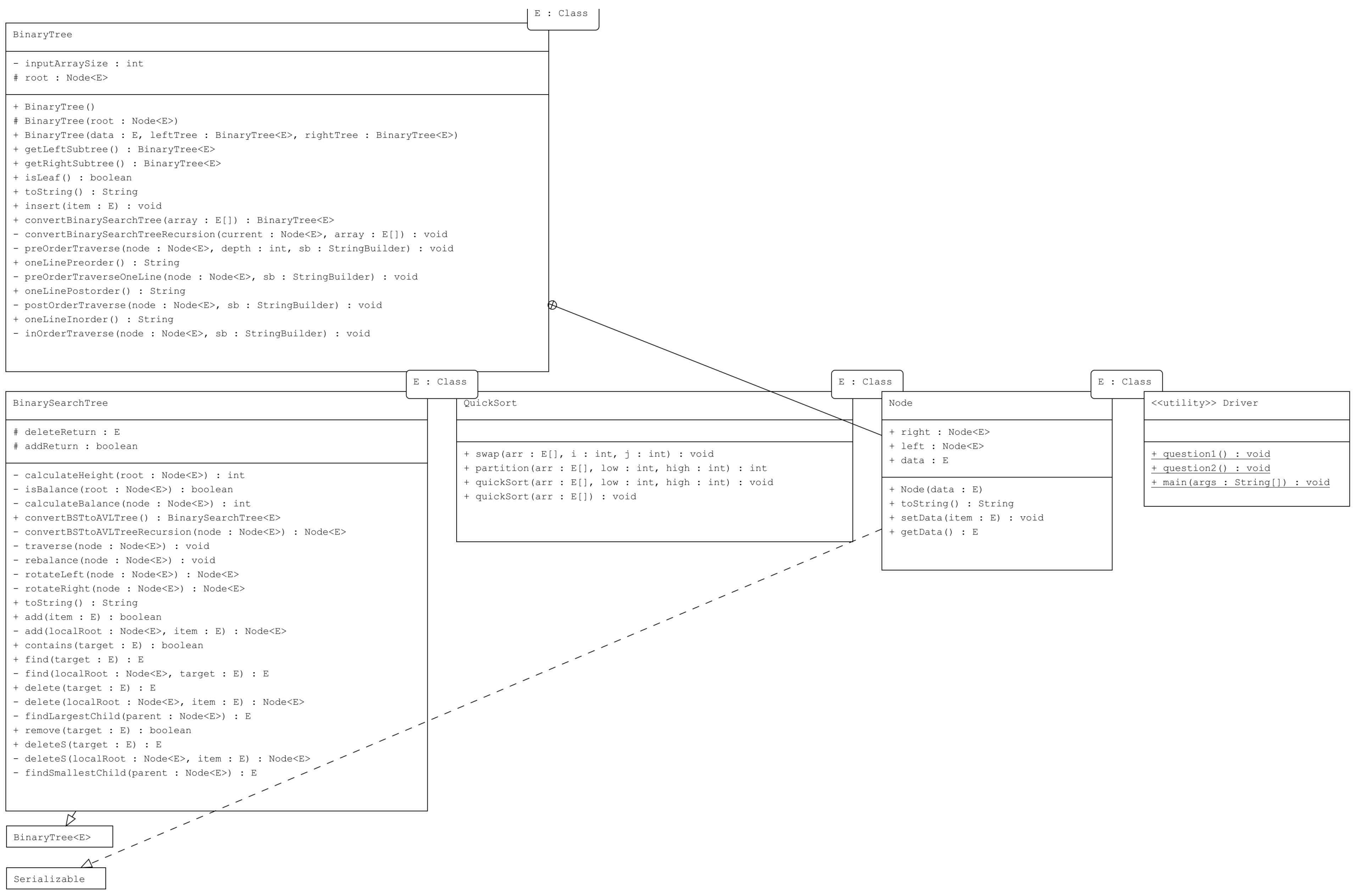
Atacan Başaran
200104004008008

1. SYSTEM REQUIREMENTS

Question 1 -> In question 1, we are expected to write a method that takes a binary tree and an array and create a new tree that looks like bst but contains binary tree as a structure and an array as an element.The method needs to fill the data of the existing tree with the elements in the array in the desired order.

Question 2 -> In question 2,we are expected to write a method that takes a binary search tree and convert it to AVL tree with rebalancing.The method should rebalance the unbalanced nodes by traversing the tree.

2. CLASS DIAGRAM


E : Class
BinaryTree
- inputArraySize : int
# root : Node<E>
+ BinaryTree()
# BinaryTree(root : Node<E>)
+ BinaryTree(data : E, leftTree : BinaryTree<E>, rightTree : BinaryTree<E>)
+ getLeftSubtree() : BinaryTree<E>
+ getRightSubtree() : BinaryTree<E>
+ isLeaf() : boolean
+ toString() : String
+ insert(item : E) : void
+ convertBinarySearchTree(array : E[]) : BinaryTree<E>
- convertBinarySearchTreeRecursion(current : Node<E>, array : E[]) : void
- preOrderTraverse(node : Node<E>, depth : int, sb : StringBuilder) : void
+ oneLinePreorder() : String
- preOrderTraverseOneLine(node : Node<E>, sb : StringBuilder) : void
+ oneLinePostorder() : String
- postOrderTraverse(node : Node<E>, sb : StringBuilder) : void
+ oneLineInorder() : String
- inOrderTraverse(node : Node<E>, sb : StringBuilder) : void
E : Class
BinarySearchTree
# deleteReturn : E
# addReturn : boolean
- calculateHeight(root : Node<E>) : int
- isBalance(root : Node<E>) : boolean
- calculateBalance(node : Node<E>) : int
+ convertBSTtoAVLTree() : BinarySearchTree<E>
- convertBSTtoAVLTreeRecursion(node : Node<E>) : Node<E>
- traverse(node : Node<E>) : void
- rebalance(node : Node<E>) : void
- rotateLeft(node : Node<E>) : Node<E>
- rotateRight(node : Node<E>) : Node<E>
+ toString() : String
+ add(item : E) : boolean
- add(localRoot : Node<E>, item : E) : Node<E>
+ contains(target : E) : boolean
+ find(target : E) : E
- find(localRoot : Node<E>, target : E) : E
+ delete(target : E) : E
- delete(localRoot : Node<E>, item : E) : Node<E>
- findLargestChild(parent : Node<E>) : E
+ remove(target : E) : boolean
+ deleteS(target : E) : E
- deleteS(localRoot : Node<E>, item : E) : Node<E>
- findSmallestChild(parent : Node<E>) : E
BinaryTree<E>
Serializable
QuickSort
+ swap(arr : E[], i : int, j : int) : void
+ partition(arr : E[], low : int, high : int) : int
+ quickSort(arr : E[], low : int, high : int) : void
+ quickSort(arr : E[]) : void
E : Class
Node
+ right : Node<E>
+ left : Node<E>
+ data : E
+ Node(data : E)
+ toString() : String
+ setData(item : E) : void
+ getData() : E
E : Class
<<utility>> Driver
+ question1() : void
+ question2() : void
+ main(args : String[]) : void

3. PROBLEM SOLUTION APPROACH

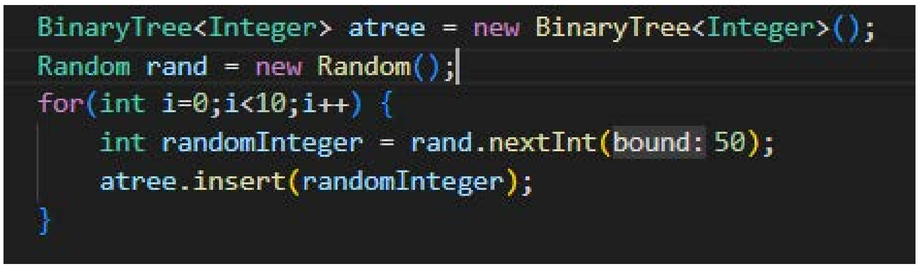
Question 1 -> In Question1, i added insert method into binary tree for testing and sort the array with quicksort for fast sorting.convertBinarySearchTree method traverses all nodes of the tree(rightTree-currentNode - leftTre) and adding the appropriate element from the sorted array to the tree's node.With this way the binary tree becomes a binary search tree.

Question 2 -> In Question2, i created calculateheight() isbalanced() methods to check balance status of current Binary search tree. convertBSTtoAVLTree method traverses all nodes of the tree and if finds unbalanced node, balance it with rebalance method(inside of it there is rotateleft-rotateright).With this way the binary search tree becomes a AVL tree.

4. TEST CASES

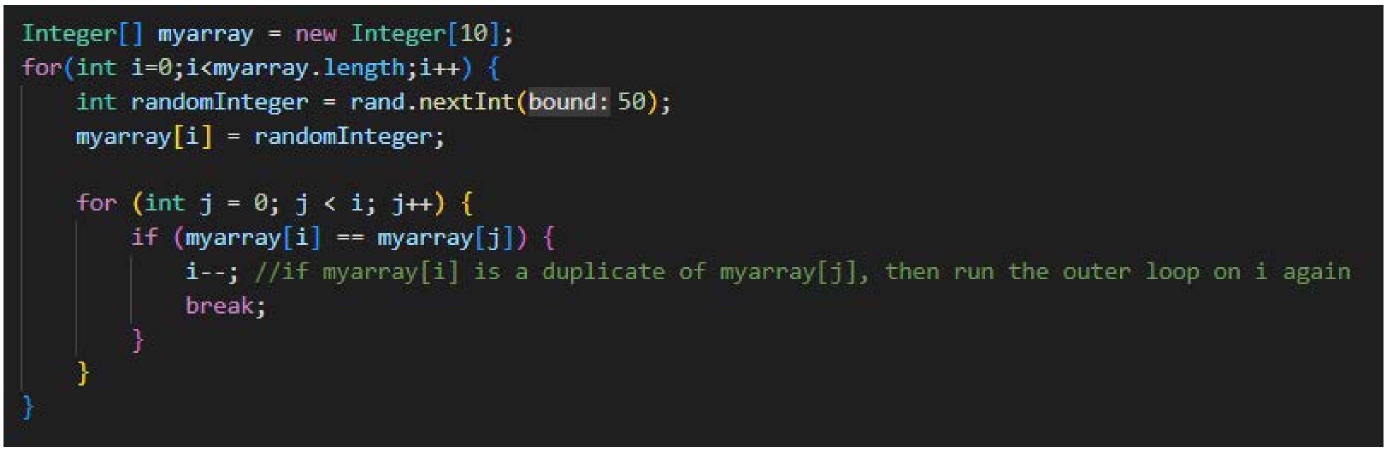
Question 1

Creating Binary Tree and filling it randomly



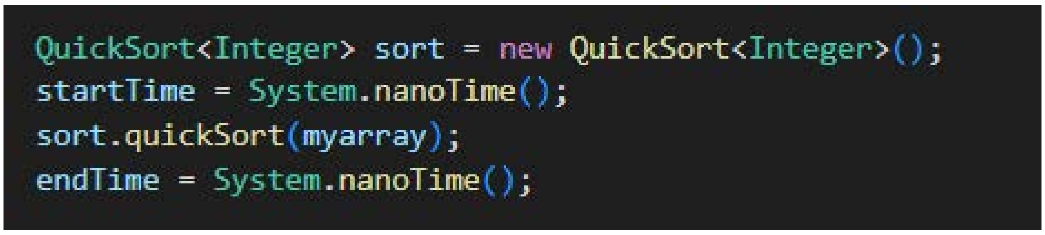
```java
BinaryTree<Integer> atree = new BinaryTree<Integer>();
Random rand = new Random();
for(int i=0;i<10;i++) {
    int randomInteger = rand.nextInt(bound: 50);
    atree.insert(randomInteger);
}
```

Creating Integer Array and filling it randomly



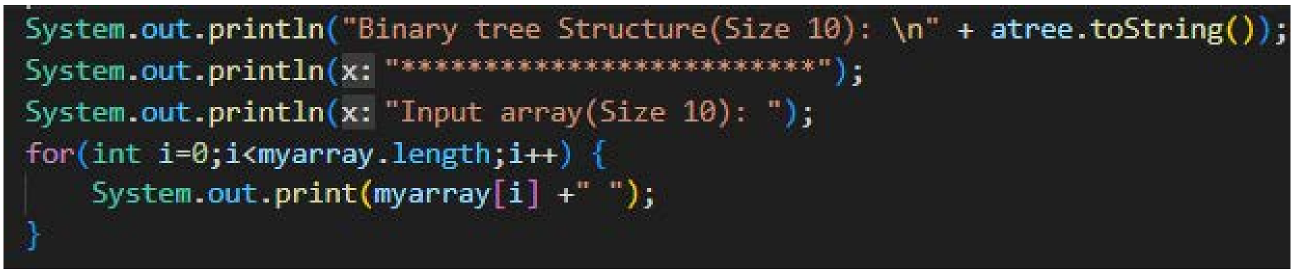
```java
Integer[] myarray = new Integer[10];
for(int i=0;i<myarray.length;i++) {
    int randomInteger = rand.nextInt(bound: 50);
    myarray[i] = randomInteger;

    for (int j = 0; j < i; j++) {
        if (myarray[i] == myarray[j]) {
            i--; //if myarray[i] is a duplicate of myarray[j], then run the outer loop on i again
            break;
        }
    }
}
```

Sorting the array with quicksort and also measure run-time



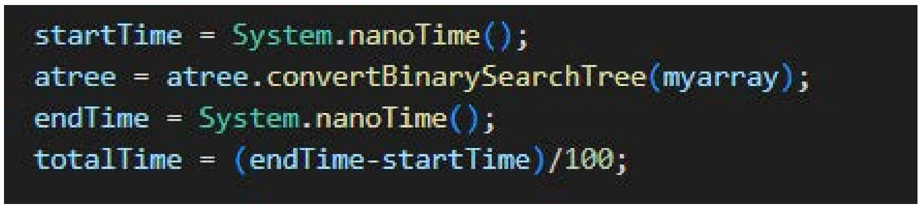
```java
QuickSort<Integer> sort = new QuickSort<Integer>();
startTime = System.nanoTime();
sort.quickSort(myarray);
endTime = System.nanoTime();
```

Printing created Binary Tree and Sorted Array



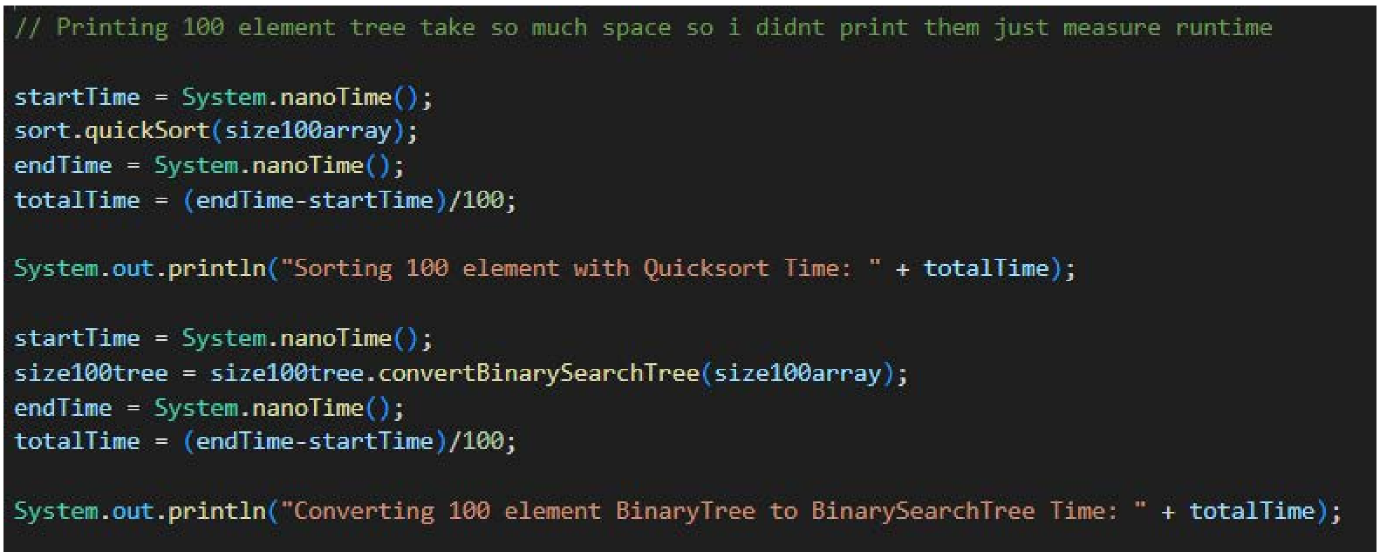
```java
System.out.println("Binary tree Structure(Size 10): \n" + atree.toString());
System.out.println(x: "************************");
System.out.println(x: "Input array(Size 10): ");
for(int i=0;i<myarray.length;i++) {
    System.out.print(myarray[i] +" ");
}
```

Convert BinaryTree to BinarySearchTree and measure run-time



```java
startTime = System.nanoTime();
atree = atree.convertBinarySearchTree(myarray);
endTime = System.nanoTime();
totalTime = (endTime-startTime)/100;
```

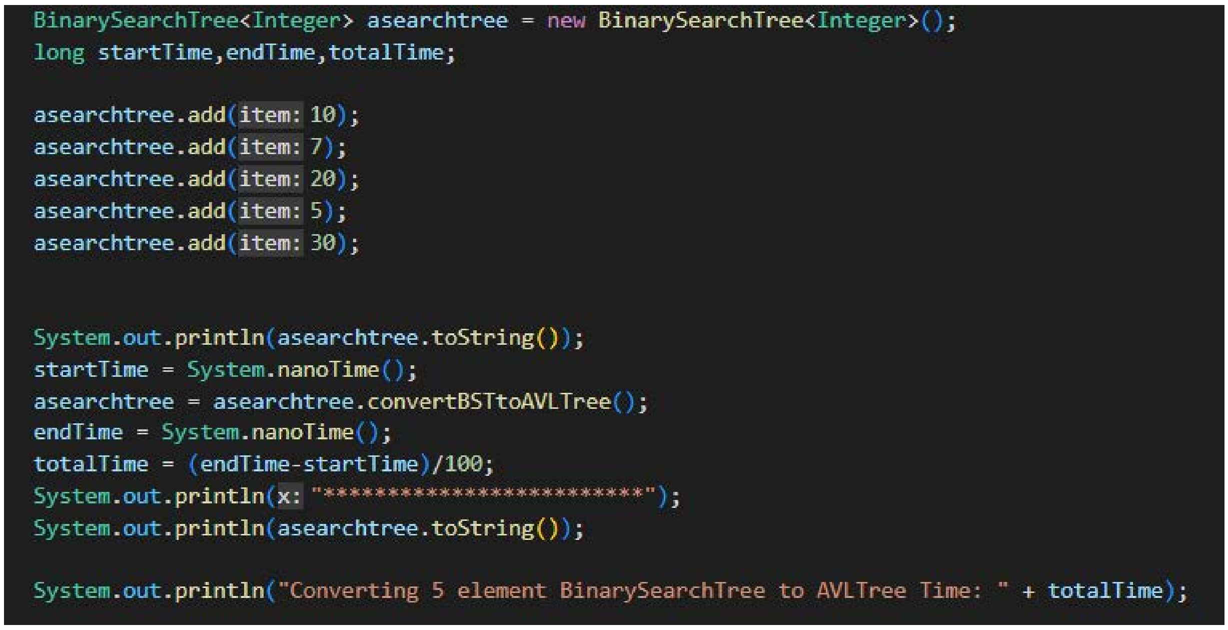
Doing same thing with 100 size



```java
// Printing 100 element tree take so much space so i didnt print them just measure runtime

startTime = System.nanoTime();
sort.quickSort(size100array);
endTime = System.nanoTime();
totalTime = (endTime-startTime)/100;

System.out.println("Sorting 100 element with Quicksort Time: " + totalTime);

startTime = System.nanoTime();
size100tree = size100tree.convertBinarySearchTree(size100array);
endTime = System.nanoTime();
totalTime = (endTime-startTime)/100;

System.out.println("Converting 100 element BinaryTree to BinarySearchTree Time: " + totalTime);
```

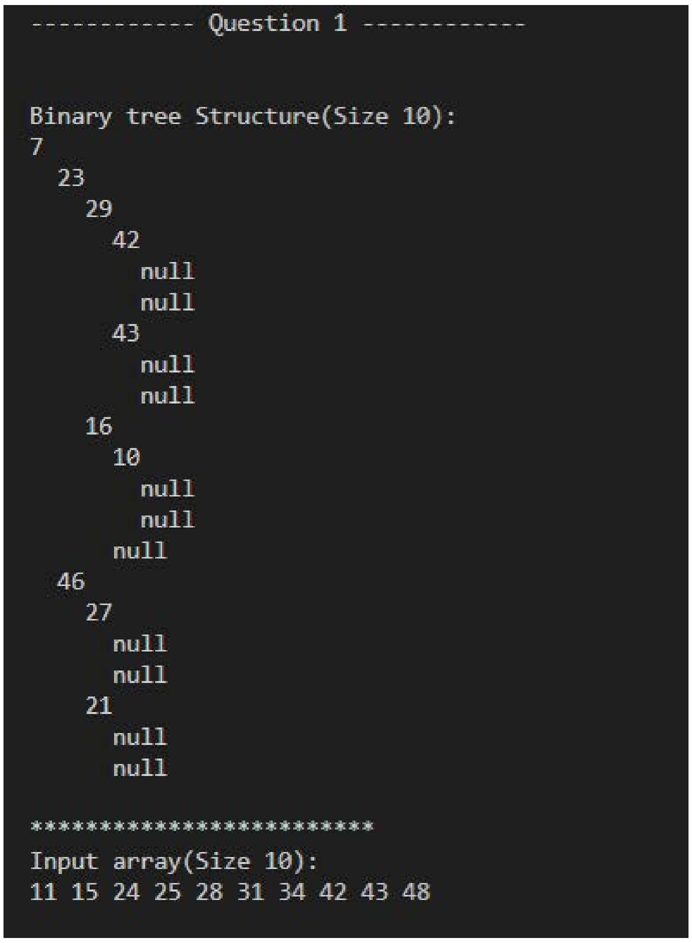
Question 2

Creating a Binary Search Tree and fill with 5 elements and convert it to AVL tree and also measure run-time

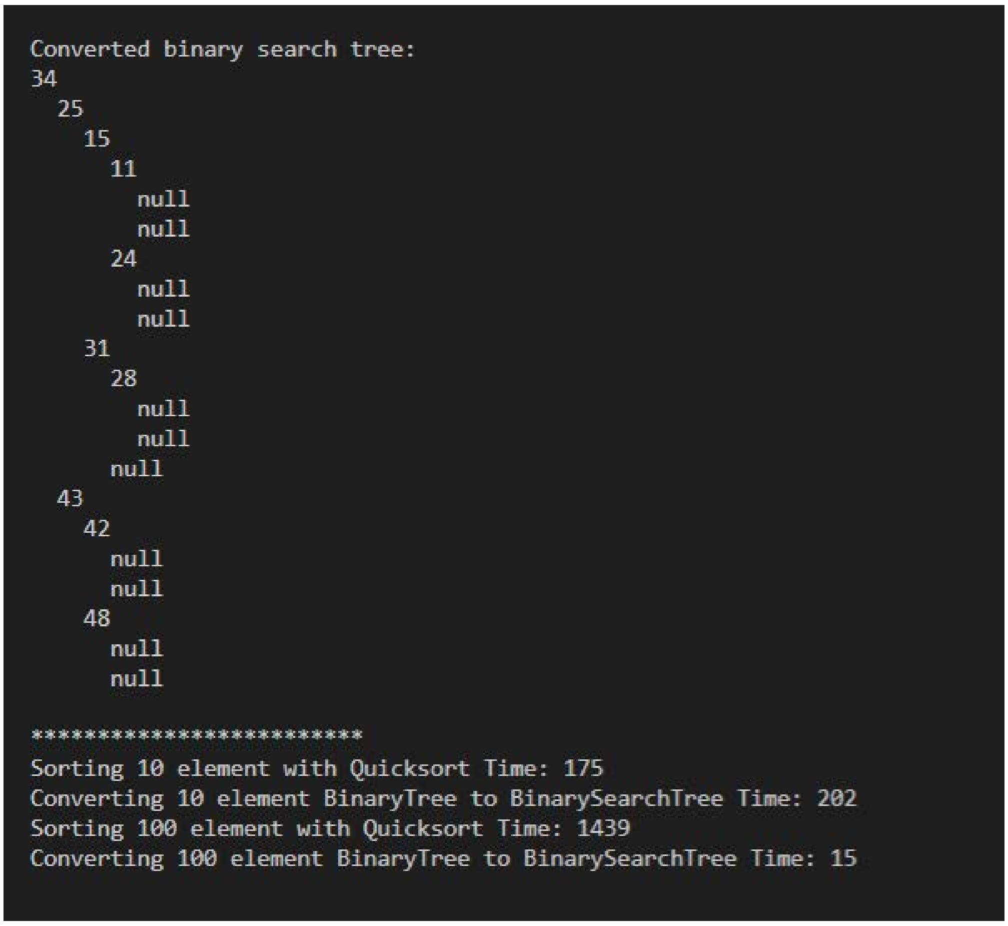


```java
BinarySearchTree<Integer> asearchtree = new BinarySearchTree<Integer>();
long startTime,endTime,totalTime;

asearchtree.add(item: 10);
asearchtree.add(item: 7);
asearchtree.add(item: 20);
asearchtree.add(item: 5);
asearchtree.add(item: 30);


System.out.println(asearchtree.toString());
startTime = System.nanoTime();
asearchtree = asearchtree.convertBSTtoAVLTree();
endTime = System.nanoTime();
totalTime = (endTime-startTime)/100;
System.out.println(x: "************************");
System.out.println(asearchtree.toString());

System.out.println("Converting 5 element BinarySearchTree to AVLTree Time: " + totalTime);
```

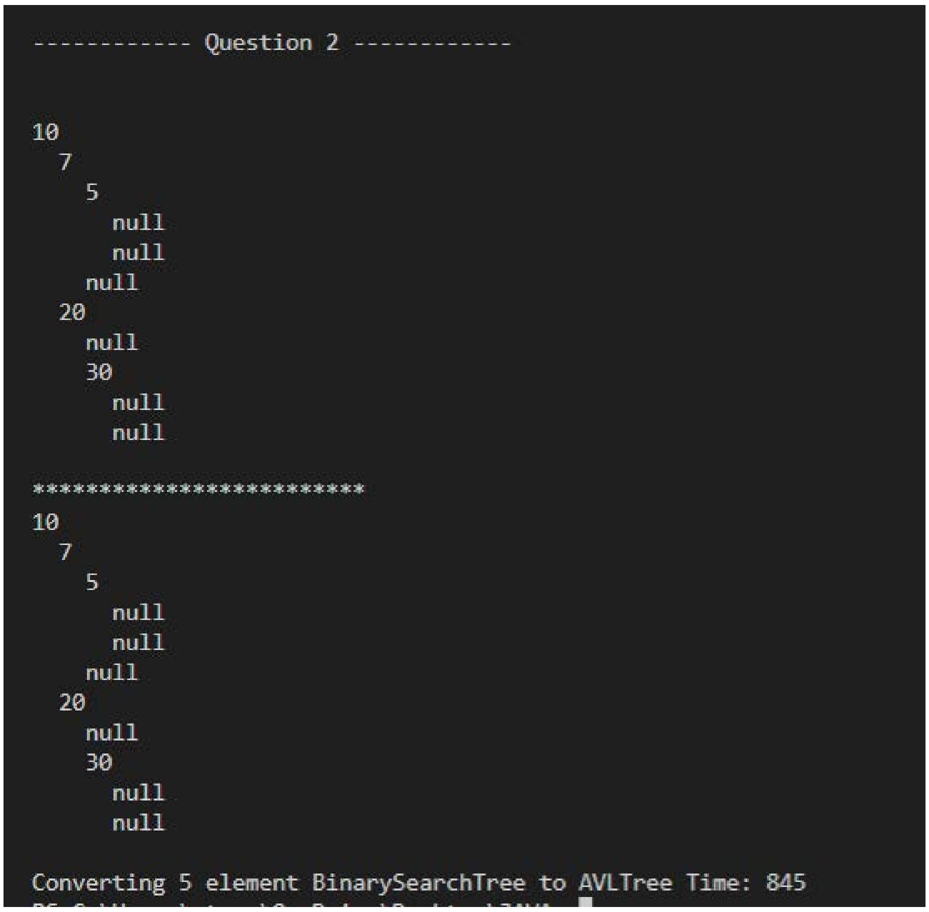
5. RUNNING AND RESULTS



```
------------ Question 1 ------------


Binary tree Structure(Size 10):
7
  23
    29
      42
        null
        null
      43
        null
        null
    16
      10
        null
        null
      null
  46
    27
      null
      null
    21
      null
      null

*************************
Input array(Size 10):
11 15 24 25 28 31 34 42 43 48
```



```
Converted binary search tree:
34
  25
    15
      11
        null
        null
      24
        null
        null
    31
      28
        null
        null
      null
  43
    42
      null
      null
    48
      null
      null

*************************
Sorting 10 element with Quicksort Time: 175
Converting 10 element BinaryTree to BinarySearchTree Time: 202
Sorting 100 element with Quicksort Time: 1439
Converting 100 element BinaryTree to BinarySearchTree Time: 15
```



```
------------ Question 2 ------------


10
  7
    5
      null
      null
    null
  20
    null
    30
      null
      null

*************************
10
  7
    5
      null
      null
    null
  20
    null
    30
      null
      null

Converting 5 element BinarySearchTree to AVLTree Time: 845
[illegible]
```

Quicksort Time complexity T(n) = θ(n*logn)

convertBinarySearchTree() Time complexity T(n) = θ(n) (Because it traverse all nodes)

convertBSTtoAVLTree() Time complexity T(n) = O(n) (Because rebalance some of node with traverse technique)