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

Ahmet Kadir Aksu
200104004114

Q1)

```
public BinaryTree<E> build (BinaryTree<E> bTree, E[] arr)
```

This method calls helper method “builder”.

```
private void builder (BinaryTree<E> bTree, E[] arr)
```

Recursive method.
Tree has n nodes, then each node is visited only once in in order traversal and hence the complexity is O(n).

Q2)

```
public static <T extends Comparable<T>> void bstToAvl
(BinarySearchTree<T> bsTree)
```

This method calls helper method “bstToAvl” .

```
private static <T> Node<T> bstToAvl(Node<T> root)
```

This method traverses the tree and would work O(n) time, but it also calls heightOf() method each time so,

```
private static <T> int heightOf(Node<T> root)
```

This is starter height method, calls helper method

```
    private static <T> int[] heightOf(Node<T> root, int[]
arr)
```

This method traverses the tree recursively and returns an array which stores heights of left and right subtrees. Thus, it is O(n).

Then bstToAvl method is O(n^2).

1. SYSTEM REQUIREMENTS

For the Q1.

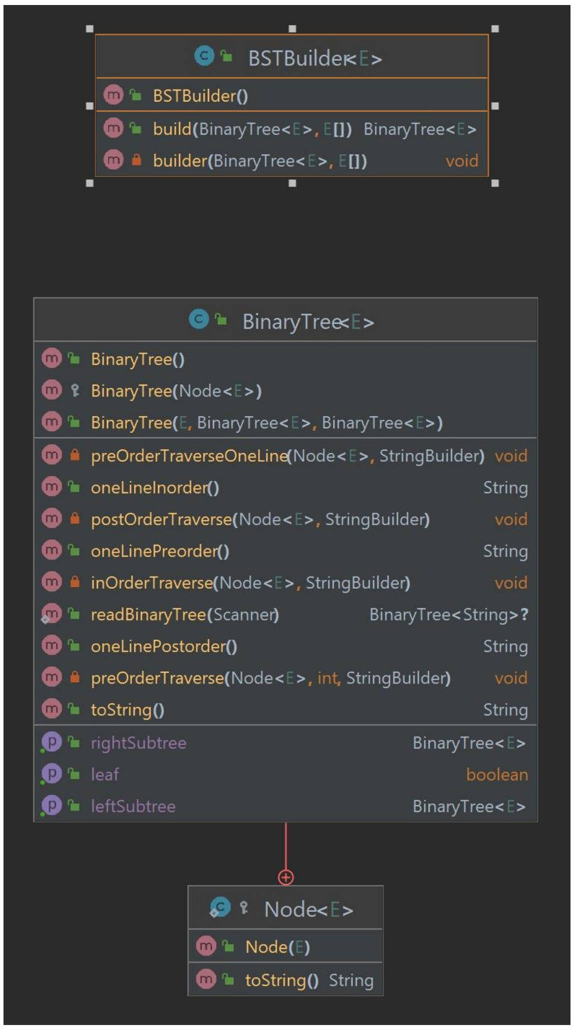
BSTBuilder Class implements the build method which takes a binary tree and an array of E type and returns a binary search tree.
To create and use a binary tree, BinaryTree class needs to be implemented. It is taken from our course book.

For the Q2.

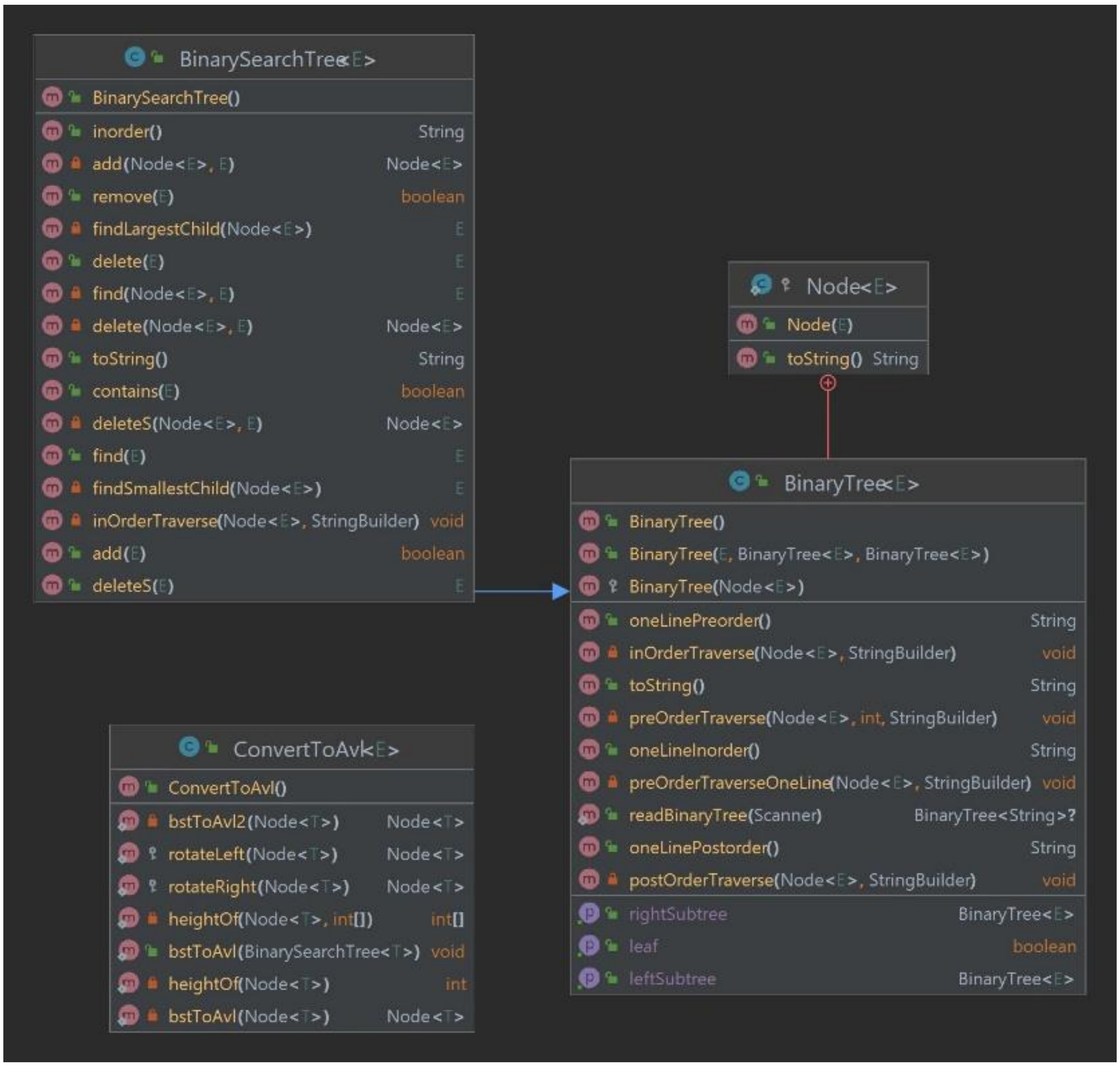
ConvertToAvl Class implements the bstToAvl method which takes a binary search tree (BST) as a parameter and prints the AVL tree obtained by rearranging the BST.
To Create and use a binary search tree, BinaryTree and BinarySearchTree classes needs to be implemented. These are taken from our course books implementation.

2. CLASS DIAGRAMS

Q1


BSTBuilder<E>
BSTBuilder()
build(BinaryTree<E>, E[]) BinaryTree<E>
builder(BinaryTree<E>, E[]) void
BinaryTree<E>
BinaryTree()
BinaryTree(Node<E>)
BinaryTree(E, BinaryTree<E>, BinaryTree<E>)
preOrderTraverseOneLine(Node<E>, StringBuilder) void
oneLineInorder() String
postOrderTraverse(Node<E>, StringBuilder) void
oneLinePreorder() String
inOrderTraverse(Node<E>, StringBuilder) void
readBinaryTree(Scanner) BinaryTree<String>?
oneLinePostorder() String
preOrderTraverse(Node<E>, int, StringBuilder) void
toString() String
rightSubtree BinaryTree<E>
leaf boolean
leftSubtree BinaryTree<E>
Node<E>
Node(E)
toString() String

Q2


BinarySearchTree<E>
BinarySearchTree()
inorder() String
add(Node<E>, E) Node<E>
remove(E) boolean
findLargestChild(Node<E>) E
delete(E) E
find(Node<E>, E) E
delete(Node<E>, E) Node<E>
toString() String
contains(E) boolean
deleteS(Node<E>, E) Node<E>
find(E) E
findSmallestChild(Node<E>) E
inOrderTraverse(Node<E>, StringBuilder) void
add(E) boolean
deleteS(E) E
Node<E>
Node(E)
toString() String
BinaryTree<E>
BinaryTree()
BinaryTree(E, BinaryTree<E>, BinaryTree<E>)
BinaryTree(Node<E>)
oneLinePreorder() String
inOrderTraverse(Node<E>, StringBuilder) void
toString() String
preOrderTraverse(Node<E>, int, StringBuilder) void
oneLineInorder() String
preOrderTraverseOneLine(Node<E>, StringBuilder) void
readBinaryTree(Scanner) BinaryTree<String>?
oneLinePostorder() String
postOrderTraverse(Node<E>, StringBuilder) void
rightSubtree BinaryTree<E>
leaf boolean
leftSubtree BinaryTree<E>
ConvertToAvl<E>
ConvertToAvl()
bstToAvl2(Node<T>) Node<T>
rotateLeft(Node<T>) Node<T>
rotateRight(Node<T>) Node<T>
heightOf(Node<T>, int[]) int[]
bstToAvl(BinarySearchTree<T>) void
heightOf(Node<T>) int
bstToAvl(Node<T>) Node<T>

3. PROBLEM SOLUTION APPROACH

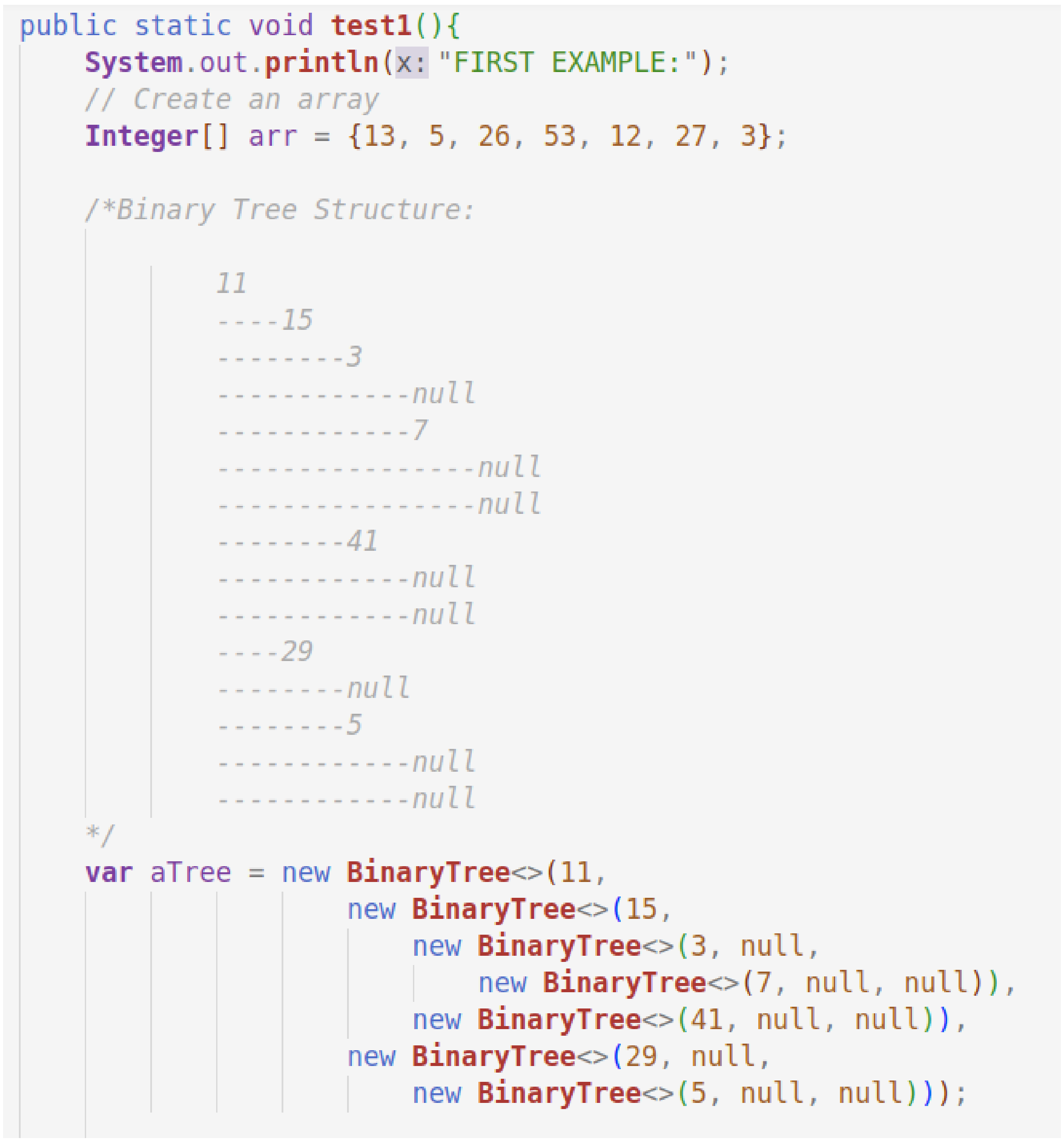
Q1- To create a binary search tree with the given binary and with the given array, first we need to sort the array and then traverse the tree in order and assign the values of the array to the nodes of the tree.

Q2- To convert binary search tree into an AVL Tree, binary search tree should be balanced. And to arrange the balance we need to use rotateLeft() and rotateRight() methods. The method traverses the binary search tree and use rotation methods if there is a need. It also uses a heightOf() method to calculate the height of a tree. If the difference between height of the left subtree and the height of the right subtree is more than 1, that means this tree is not balanced and it must be balanced.

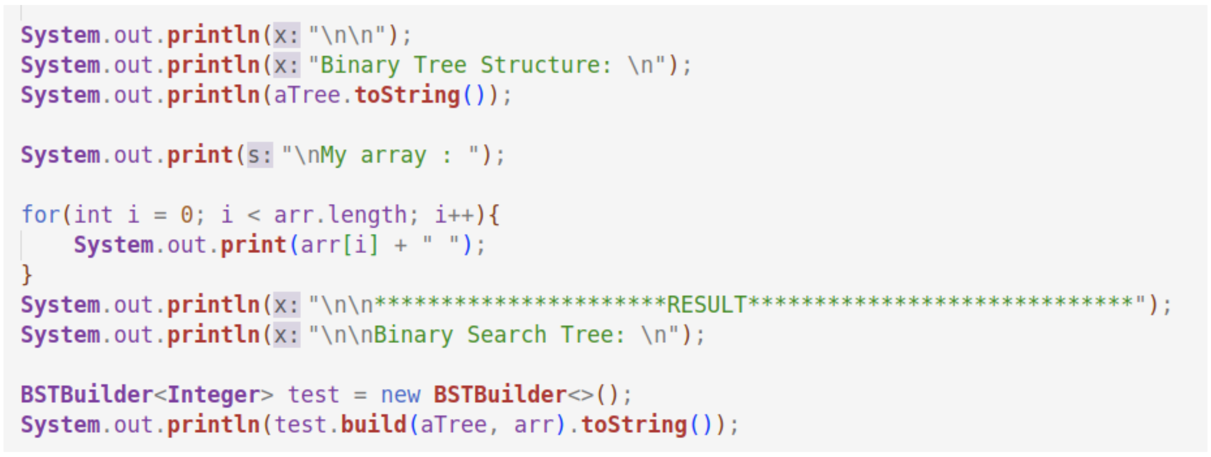
heightOf method uses a helper method that calculate the heights of the left and right subtrees recursively. This helper method returns an array of these two values. And starter method returns the greater one.

4. TEST CASES

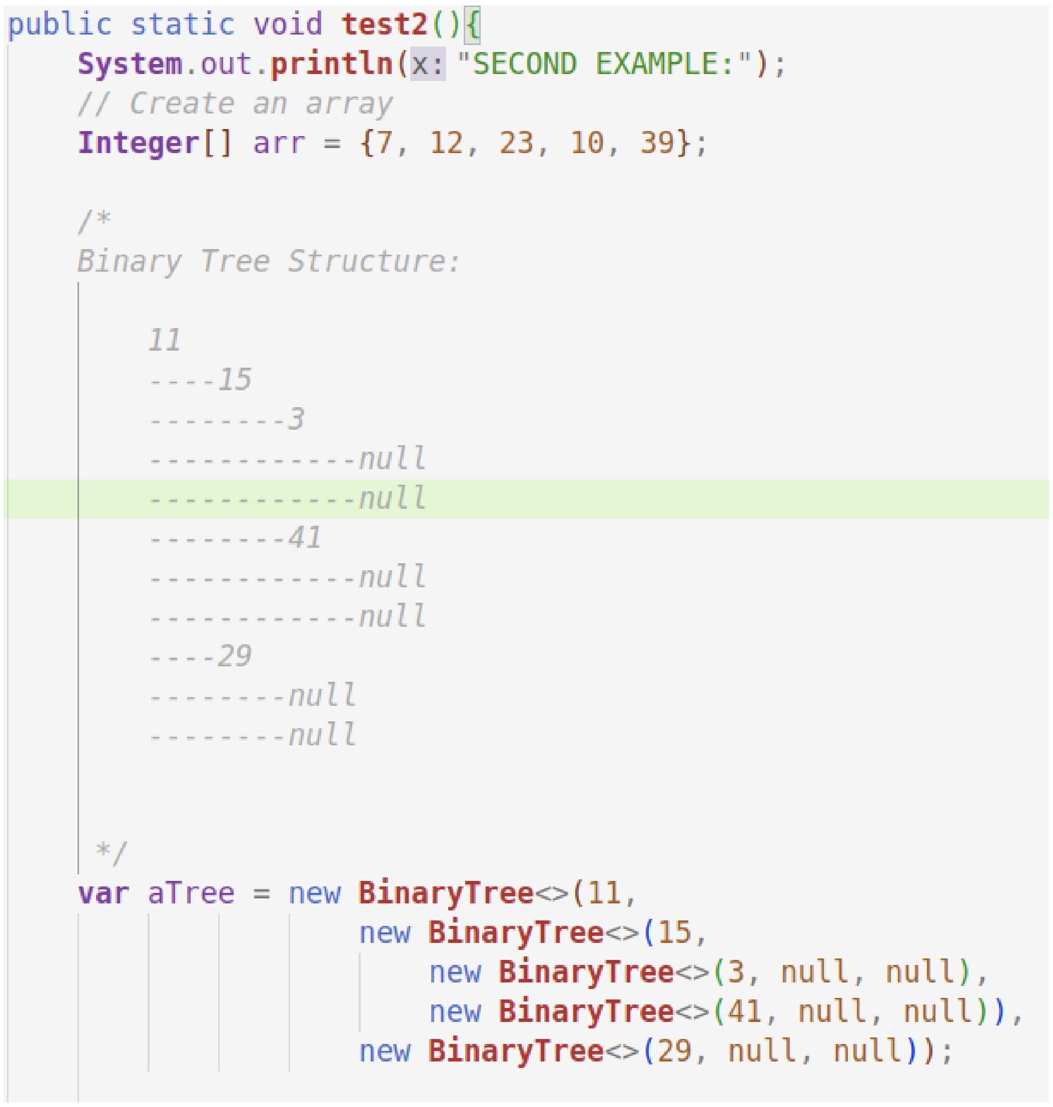
Q1 – Testing (1) BSTBuilder class, build method.

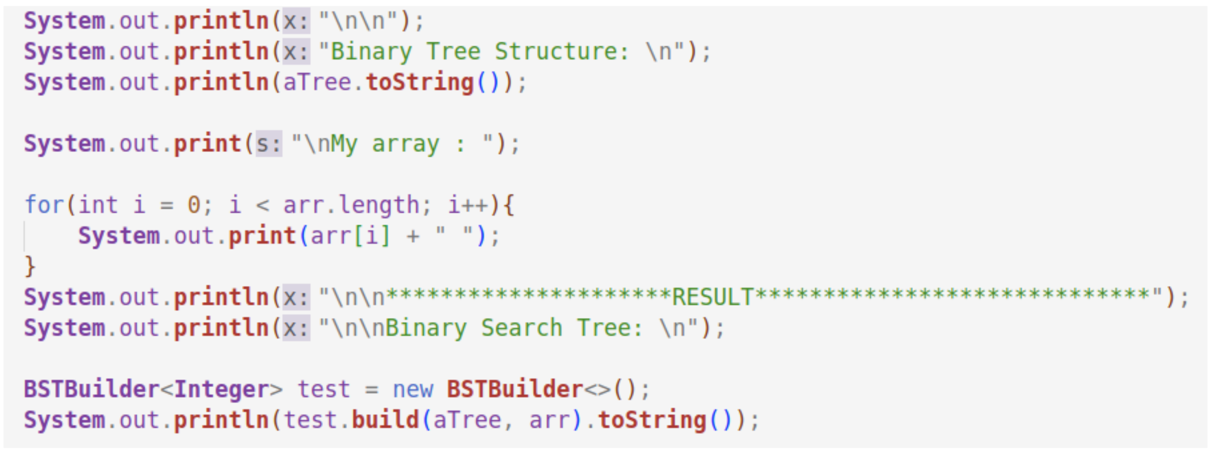


```
public static void test1(){
    System.out.println(x: "FIRST EXAMPLE:");
    // Create an array
    Integer[] arr = {13, 5, 26, 53, 12, 27, 3};

    /*Binary Tree Structure:

            11
            ----15
            --------3
            ------------null
            ------------7
            ----------------null
            ----------------null
            --------41
            ------------null
            ------------null
            ----29
            --------null
            --------5
            ------------null
            ------------null
    */
    var aTree = new BinaryTree<>(11,
                    new BinaryTree<>(15,
                        new BinaryTree<>(3, null,
                            new BinaryTree<>(7, null, null)),
                        new BinaryTree<>(41, null, null)),
                    new BinaryTree<>(29, null,
                        new BinaryTree<>(5, null, null)));
```


```
System.out.println(x: "\n\n");
System.out.println(x: "Binary Tree Structure: \n");
System.out.println(aTree.toString());

System.out.print(s: "\nMy array : ");

for(int i = 0; i < arr.length; i++){
    System.out.print(arr[i] + " ");
}
System.out.println(x: "\n\n**********************RESULT*****************************");
System.out.println(x: "\n\nBinary Search Tree: \n");

BSTBuilder<Integer> test = new BSTBuilder<>();
System.out.println(test.build(aTree, arr).toString());
```

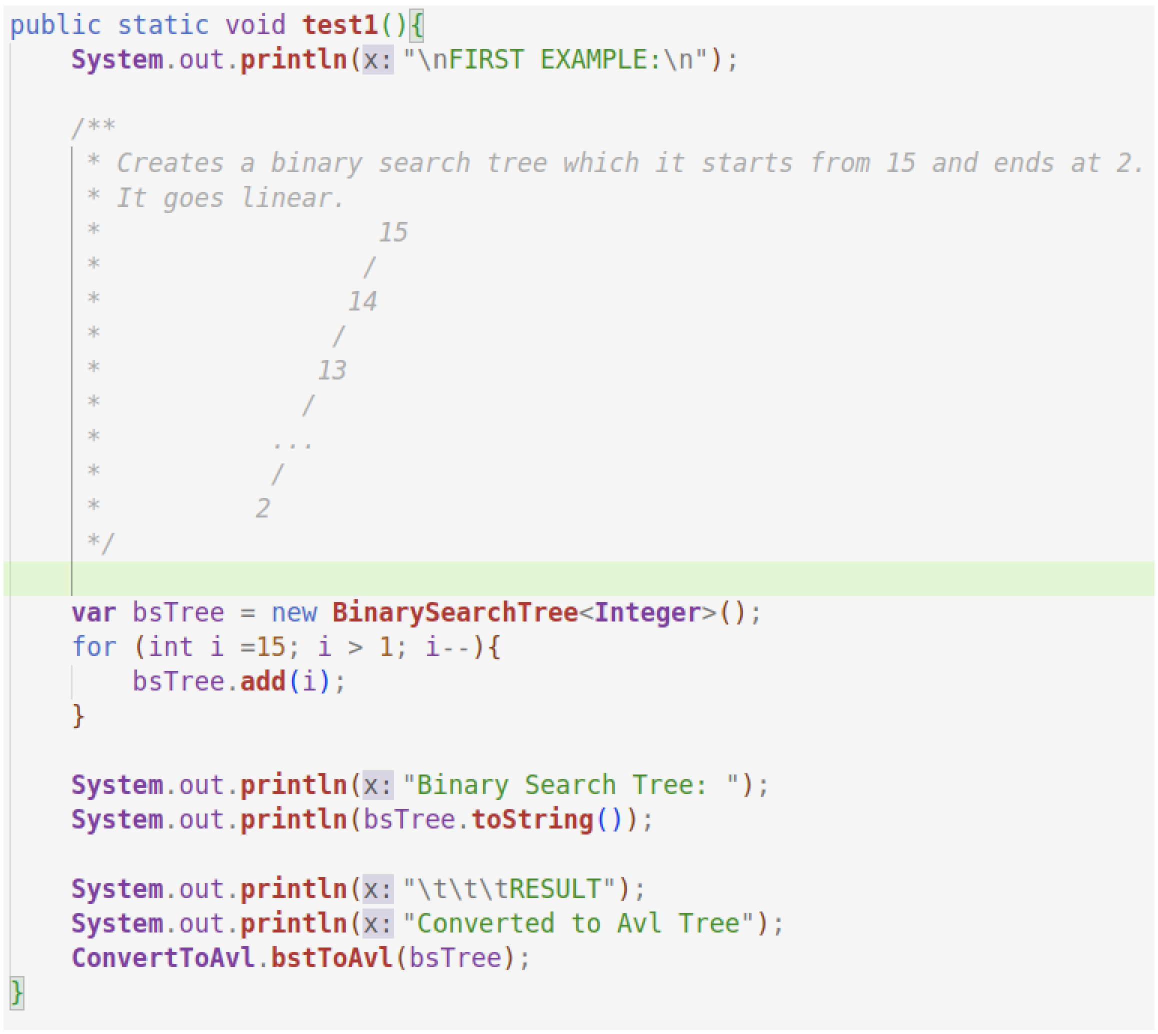
-Testing (2) BSTBuilder class, build method.


```
public static void test2(){
    System.out.println(x: "SECOND EXAMPLE:");
    // Create an array
    Integer[] arr = {7, 12, 23, 10, 39};

    /*
    Binary Tree Structure:

        11
        ----15
        --------3
        ------------null
        ------------null
        --------41
        ------------null
        ------------null
        ----29
        --------null
        --------null


     */
    var aTree = new BinaryTree<>(11,
                        new BinaryTree<>(15,
                            new BinaryTree<>(3, null, null),
                            new BinaryTree<>(41, null, null)),
                        new BinaryTree<>(29, null, null));
```



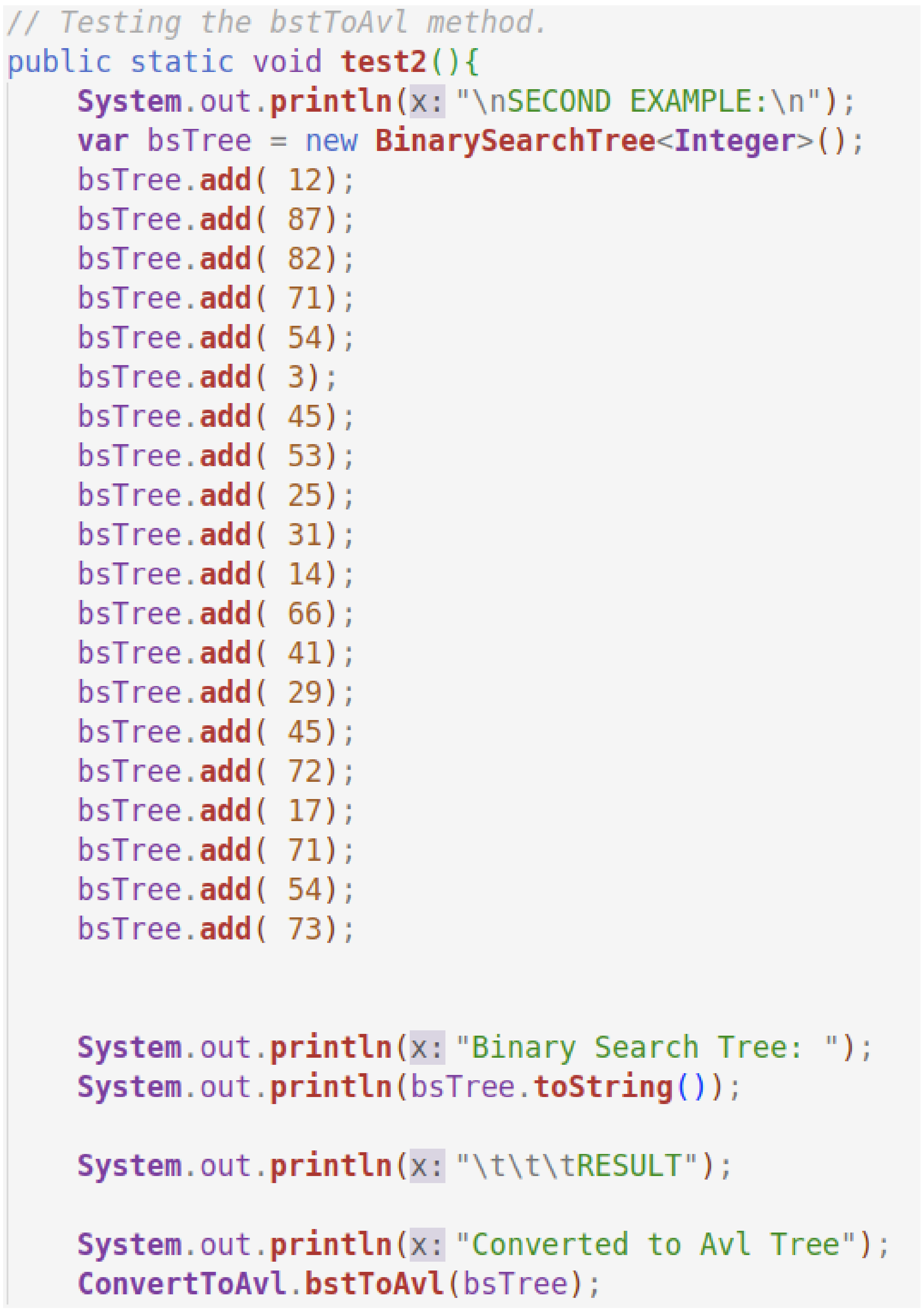
```java
System.out.println(x: "\n\n");
System.out.println(x: "Binary Tree Structure: \n");
System.out.println(aTree.toString());

System.out.print(s: "\nMy array : ");

for(int i = 0; i < arr.length; i++){
    System.out.print(arr[i] + " ");
}
System.out.println(x: "\n\n*********************RESULT*****************************");
System.out.println(x: "\n\nBinary Search Tree: \n");

BSTBuilder<Integer> test = new BSTBuilder<>();
System.out.println(test.build(aTree, arr).toString());
```

Q2 – Testing (1) ConvertToAvl class, bstToAvl method.



```java
public static void test1(){
    System.out.println(x: "\nFIRST EXAMPLE:\n");

    /**
     * Creates a binary search tree which it starts from 15 and ends at 2.
     * It goes linear.
     *                  15
     *                 /
     *                14
     *               /
     *              13
     *             /
     *           ...
     *           /
     *          2
     */

    var bsTree = new BinarySearchTree<Integer>();
    for (int i =15; i > 1; i--){
        bsTree.add(i);
    }

    System.out.println(x: "Binary Search Tree: ");
    System.out.println(bsTree.toString());

    System.out.println(x: "\t\t\tRESULT");
    System.out.println(x: "Converted to Avl Tree");
    ConvertToAvl.bstToAvl(bsTree);
}
```

Testing (2) ConvertToAvl class, bstToAvl method.

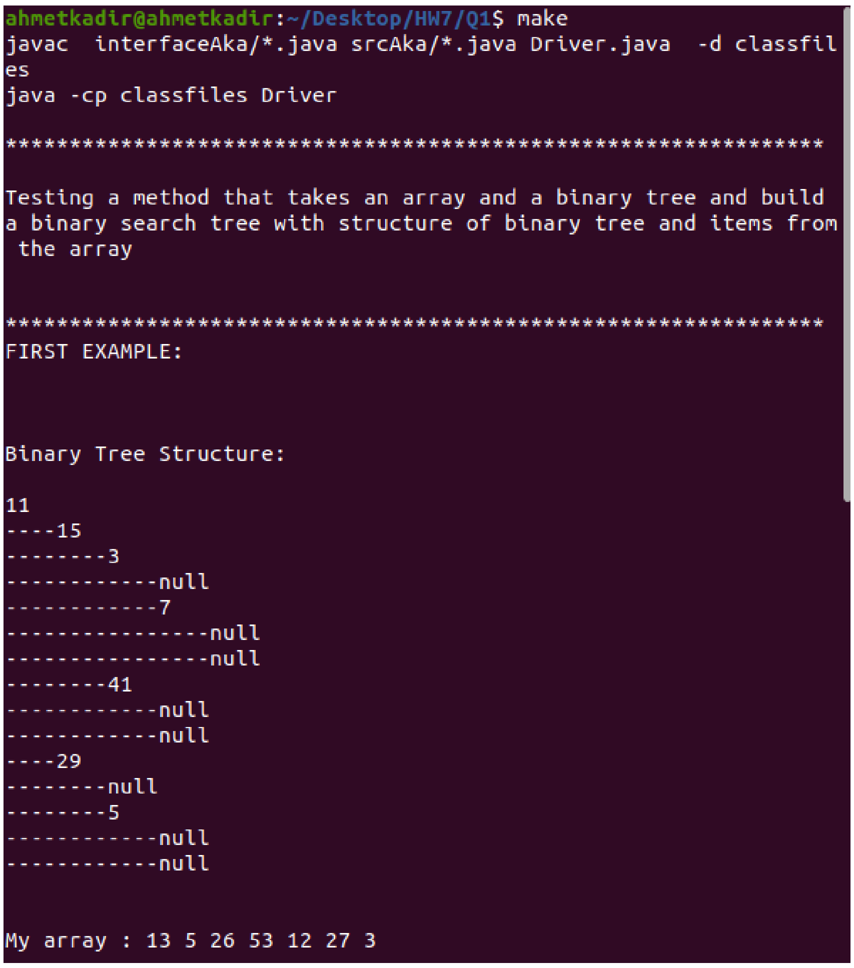


```java
// Testing the bstToAvl method.
public static void test2(){
    System.out.println(x: "\nSECOND EXAMPLE:\n");
    var bsTree = new BinarySearchTree<Integer>();
    bsTree.add( 12);
    bsTree.add( 87);
    bsTree.add( 82);
    bsTree.add( 71);
    bsTree.add( 54);
    bsTree.add( 3);
    bsTree.add( 45);
    bsTree.add( 53);
    bsTree.add( 25);
    bsTree.add( 31);
    bsTree.add( 14);
    bsTree.add( 66);
    bsTree.add( 41);
    bsTree.add( 29);
    bsTree.add( 45);
    bsTree.add( 72);
    bsTree.add( 17);
    bsTree.add( 71);
    bsTree.add( 54);
    bsTree.add( 73);


    System.out.println(x: "Binary Search Tree: ");
    System.out.println(bsTree.toString());

    System.out.println(x: "\t\t\tRESULT");

    System.out.println(x: "Converted to Avl Tree");
    ConvertToAvl.bstToAvl(bsTree);
```

5- RUNNING AND RESULTS

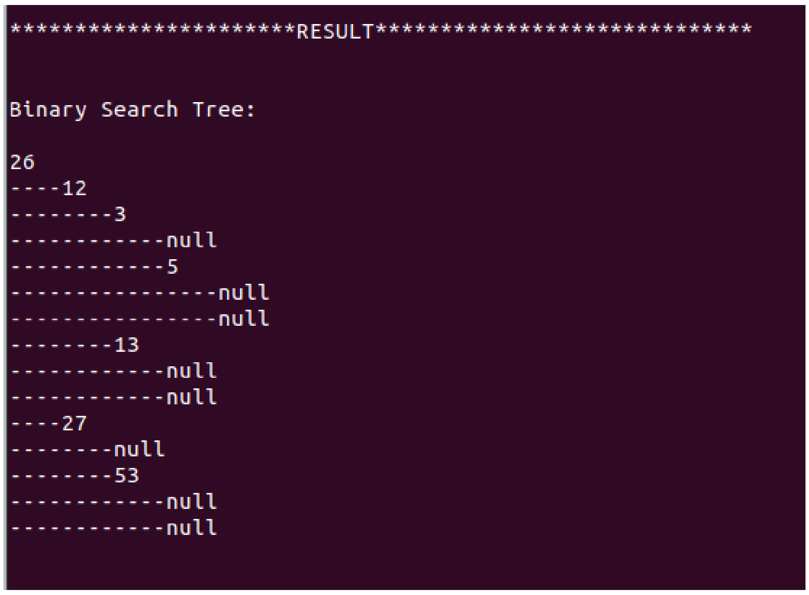
Q1 result:

First example:



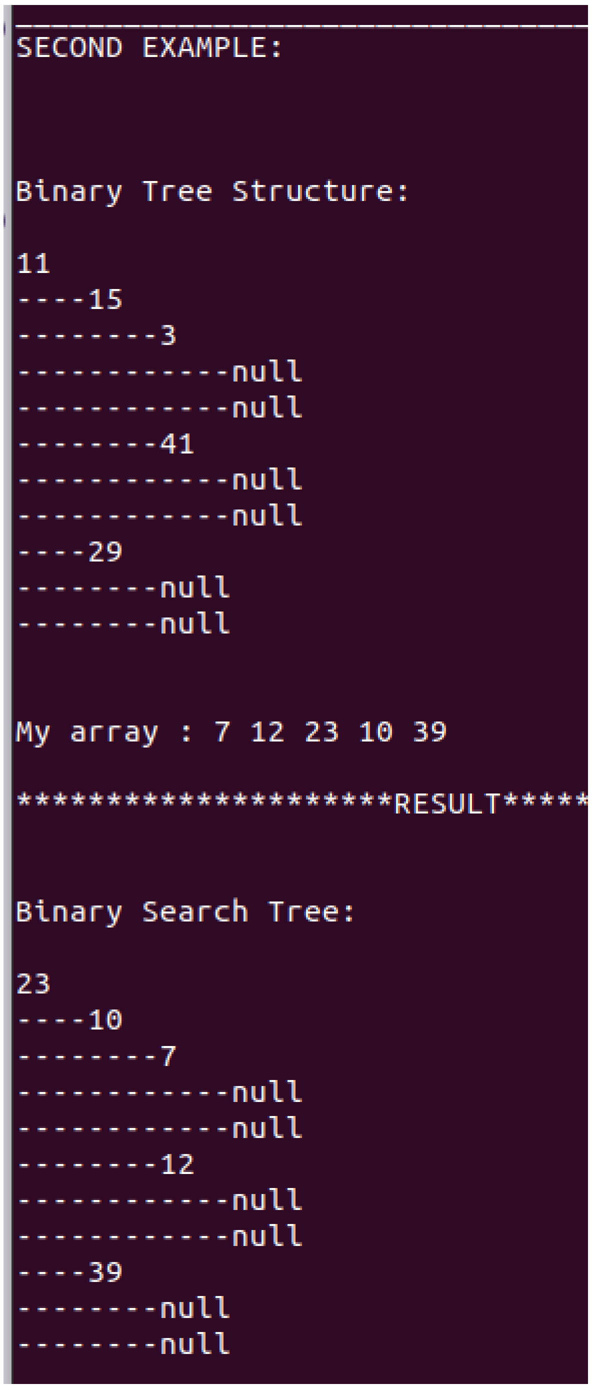
```
ahmetkadir@ahmetkadir:~/Desktop/HW7/Q1$ make
javac  interfaceAka/*.java srcAka/*.java Driver.java  -d classfil
es
java -cp classfiles Driver

****************************************************************

Testing a method that takes an array and a binary tree and build
a binary search tree with structure of binary tree and items from
 the array


****************************************************************
FIRST EXAMPLE:



Binary Tree Structure:

11
----15
--------3
------------null
------------7
----------------null
----------------null
--------41
------------null
------------null
----29
--------null
--------5
------------null
------------null


My array : 13 5 26 53 12 27 3
```

Result of the first example:



```
**********************RESULT******************************

Binary Search Tree:

26
----12
--------3
------------null
------------5
----------------null
----------------null
--------13
------------null
------------null
----27
--------null
--------53
------------null
------------null
```

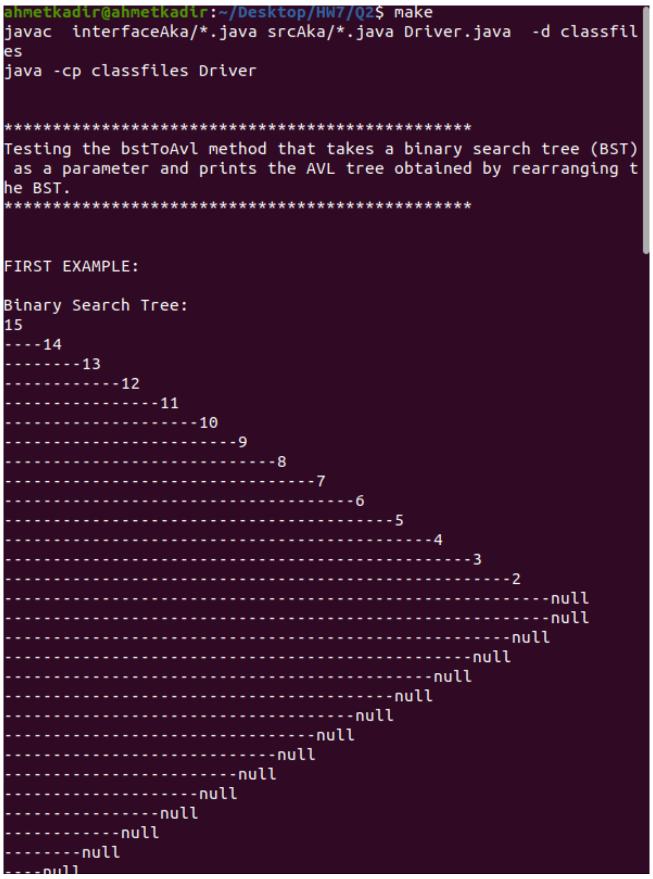
Second Example:



```
________________________________
SECOND EXAMPLE:



Binary Tree Structure:

11
----15
--------3
------------null
------------null
--------41
------------null
------------null
----29
--------null
--------null


My array : 7 12 23 10 39

*********************RESULT*****


Binary Search Tree:

23
----10
--------7
------------null
------------null
--------12
------------null
------------null
----39
--------null
--------null
```

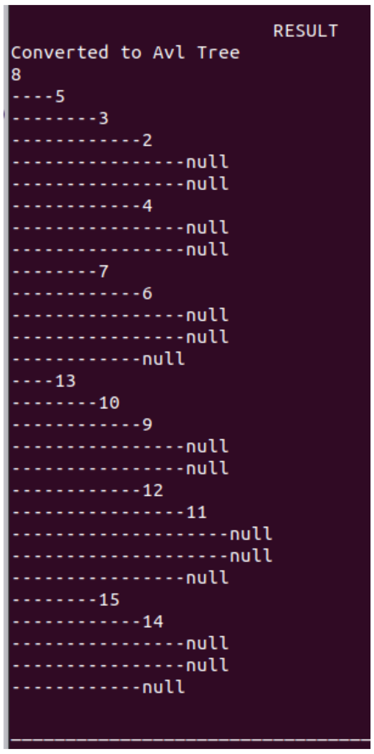
Q2 result:

First Example:



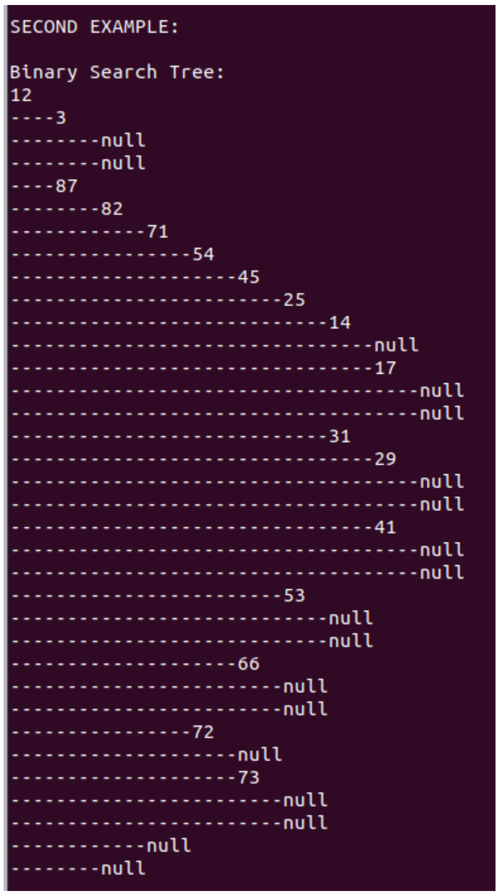
```
ahmetkadir@ahmetkadir:~/Desktop/HW7/Q2$ make
javac  interfaceAka/*.java srcAka/*.java Driver.java  -d classfil
es
java -cp classfiles Driver


************************************************
Testing the bstToAvl method that takes a binary search tree (BST)
 as a parameter and prints the AVL tree obtained by rearranging t
he BST.
************************************************


FIRST EXAMPLE:

Binary Search Tree:
15
----14
--------13
------------12
----------------11
--------------------10
------------------------9
----------------------------8
--------------------------------7
------------------------------------6
----------------------------------------5
--------------------------------------------4
------------------------------------------------3
----------------------------------------------------2
--------------------------------------------------------null
--------------------------------------------------------null
----------------------------------------------------null
------------------------------------------------null
--------------------------------------------null
----------------------------------------null
------------------------------------null
--------------------------------null
----------------------------null
------------------------null
--------------------null
----------------null
------------null
--------null
----null
```

Result of the first example:



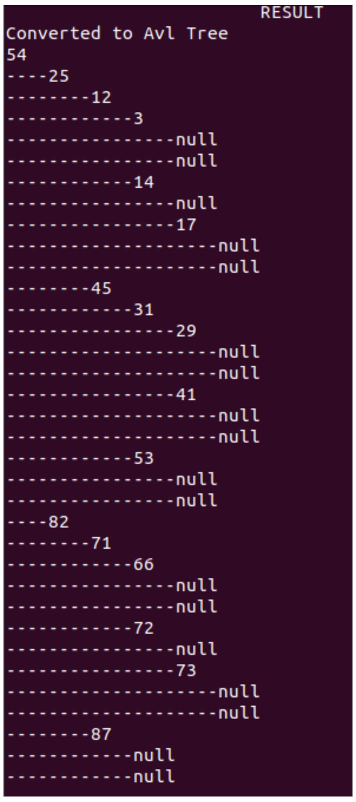
```
                           RESULT
Converted to Avl Tree
8
----5
--------3
------------2
----------------null
----------------null
------------4
----------------null
----------------null
--------7
------------6
----------------null
----------------null
------------null
----13
--------10
------------9
----------------null
----------------null
------------12
----------------11
--------------------null
--------------------null
----------------null
--------15
------------14
----------------null
----------------null
------------null
```

Second Example:



```
SECOND EXAMPLE:

Binary Search Tree:
12
----3
--------null
--------null
----87
--------82
------------71
----------------54
--------------------45
------------------------25
----------------------------14
--------------------------------null
--------------------------------17
------------------------------------null
------------------------------------null
----------------------------31
--------------------------------29
------------------------------------null
------------------------------------null
--------------------------------41
------------------------------------null
------------------------------------null
------------------------53
----------------------------null
----------------------------null
--------------------66
------------------------null
------------------------null
----------------72
--------------------null
--------------------73
------------------------null
------------------------null
------------null
--------null
```

Result of the second example:



```
                              RESULT
Converted to Avl Tree
54
----25
--------12
------------3
----------------null
----------------null
------------14
----------------null
----------------17
--------------------null
--------------------null
--------45
------------31
----------------29
--------------------null
--------------------null
----------------41
--------------------null
--------------------null
------------53
----------------null
----------------null
----82
--------71
------------66
----------------null
----------------null
------------72
----------------null
----------------73
--------------------null
--------------------null
--------87
------------null
------------null
```