ICS-202 Lab-07 Report

Mohammed Busaleh - 202158210

Exercise 1:

Code:

```
protected void rotateRight() {
     System.out.println("RIGHT ROTATION");
   //Q1
    BSTNode<T> tempNode = root.right;
    root.right = root.left;
    root.left = root.right.left;
    root.right.left = root.right.right;
    root.right.right = tempNode;

    T val = (T) root.el;
    root.el = root.right.el;
    root.right.el = val;

    getRightAVL().adjustHeight();
    adjustHeight();
}

protected void rotateLeft() {
    System.out.println("LEFT ROTATION");
   BSTNode<T> tempNode = root.left;
   root.left = root.right;
   root.right = root.left.right;
   root.left.right = root.left.left;
   root.left.left = tempNode;

   T val = (T) root.el;
   root.el = root.left.el;
   root.left.el = val;

   getLeftAVL().adjustHeight();
   adjustHeight();
 }
```

```
 protected void rotateLeftRight()
{
   System.out.println("Double Rotation...");
   //Q1
    getLeftAVL().rotateLeft();
    getLeftAVL().adjustHeight();
    this.rotateRight();
    this.adjustHeight();
}


protected void rotateRightLeft()
{
     System.out.println("Double Rotation...");
   getRightAVL().rotateRight();
   getRightAVL().adjustHeight();
   this.rotateLeft();
   this.adjustHeight();
}
```

```
public void deleteAVL(T el) {
  //Q1
    if(!isInTree(el)){
        System.out.println(el + " is not in the list");
        return;
    }

    super.deleteByCopying(el);
    this.balance();
}
```

Exercise 2:

Code:

```
System.out.println("\n\nExercise 2");
AVLTree<Integer> numAVL = new AVLTree<>();
int[] values = {8, 14, 12, 18, 23, 20, 15, 13, 7, 16};
for (int i : values)
    numAVL.insertAVL(i);

numAVL.breadthFirst();

System.out.print("\n\nEnter Three elements to be deleted for the AVL: ");
int el1 = inp.nextInt(), el2 = inp.nextInt(), el3 = inp.nextInt();
numAVL.deleteAVL(el1);
numAVL.deleteAVL(el2);
numAVL.deleteAVL(el3);
System.out.print("The AVL after deleting the elements: ");
numAVL.breadthFirst();
```

Exercise 3:

Code:

```
System.out.println("\n\nExercise 3");
AVLTree<String> wordsAVL = new AVLTree<>();
try {
    Scanner inpFile = new Scanner(new File("sampletextfile.txt"));
    while(inpFile.hasNext()){
        String word= inpFile.next();
        if(!wordsAVL.isInTree(word)){
            wordsAVL.insertAVL(word);
        }
    }
} catch (FileNotFoundException e) {
    throw new RuntimeException(e);
}
wordsAVL.inorder();
```

Output:

```
Balancing node with el: a1
LEFT ROTATION
Balancing node with el: a3
LEFT ROTATION
a2 a1 a4 a3 a5
```

```
Exercise 2
Balancing node with el: 8
Double Rotation...
RIGHT ROTATION
LEFT ROTATION
Balancing node with el: 14
LEFT ROTATION
Balancing node with el: 12
LEFT ROTATION
Balancing node with el: 18
Double Rotation...
LEFT ROTATION
RIGHT ROTATION
14 12 18 8 13 15 23 7 16 20
```

Exercise 3 sentence:

"a acts algorithms an analysis and are area as atomic both branch centuries computations decomposition derivative development devoted efficient element example expanding expedite finite for function in infinite is major matrices matrix method methods neardiagonal numerical occur of old on operator other particular planetary practically representing research series simple simplify sparse structures subject such tailored taylor that the theoretically theory to today which"