- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.
- Failure to export properly will result in your work not getting marked.

To submit:

Export your project to a JAR file, with source code.

Name your JAR file ID_Week16_Q1.jar. For example, 6623110021_Week16_Q1.jar

Submit the JAR file on MyCourseville.

Write code for class BSTRecursive. The file BSTRecursive.java is given. You must write 2 methods for this Binary Search Tree.

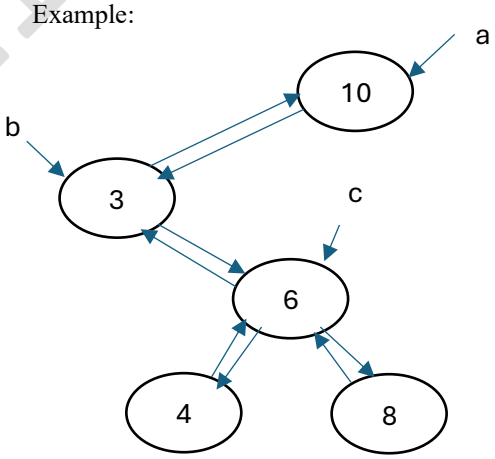
- All necessary files for Binary Search Tree are given. **Copy them into your Eclipse project**.
- You may have to use other codes from the lecture, such as code for sorting.
- **JUnit** test files are **TestSize.java** and **TestNonAVLNodes.java**.

- All your code modifications must only be in BSTRecursive.java. **All other files will not be used in marking**.
- You can write new methods in BSTRecusive.java. Make sure you write them under the specified region in the given code.

- Method to find the height of a subtree that has n as its root, **height(BSTNode n)**, is available.
- Method to check if AVL constraint holds for a single node n, **isAVL(BSTNode n)**, is available.
- Assume data in any node is >= 1.
- Assume no duplicate data are allowed in the tree.

Write the following methods:

- public int size(BSTNode n):
  - Return how many nodes a subtree has, if n is the root of the subtree.
  - Example:



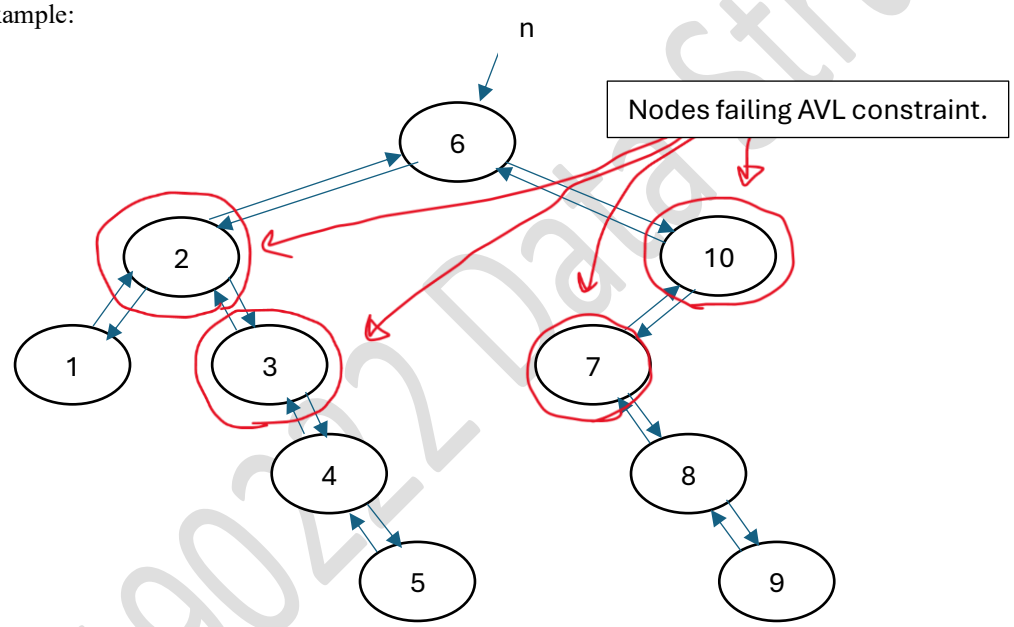
```
size(a) returns 5.

size(b) returns 4.

size(c) returns 3.
```

- public int[] nonAVLNodes(BSTNode n):
  - For a subtree that has n as its root:
    - Assume the whole tree always has at least one node, but n can be null since we can look at any node in the tree.
    - Assume the subtree does not contain more than 10 nodes.
    - This method returns the values in all nodes that do not satisfy AVL constraint.
    - The returned values are in an array. The array size is always 10.
    - The returned array must be sorted from small to large. If less than 10 nodes fail AVL constraint, the array will have some leading zero(s).
    - You can make use of variables introduced above this method (in the source code).
    - You can also add your own variables.

Example:




```
nonAVLNodes(n) returns array {0,0,0,0,0,0,2,3,7,10}
```

Scoring Criteria:

Only modify **BSTRecursive.java**. Other files must not be changed!

The total score is 10.

TestSize.java	5 marks (1 for each test case)
TestNonAVLNodes	5 marks (1 for each test case)

However, if your code does not perform (regarding asymptotic runtime) well enough, you will only get 8 marks maximum.