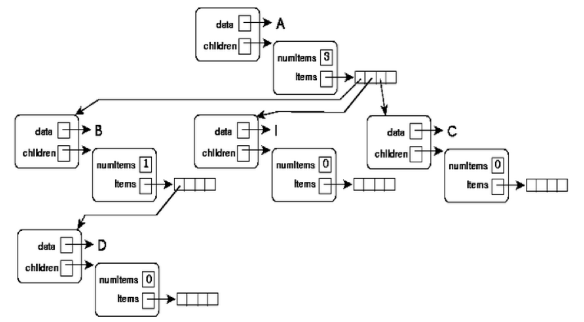
**ITE7107: Laboratory Exercise 002 Answers**

C++ Trees

Note: Source codes used for this exercise, together with this answer sheet are uploaded in the GitHub repository: <https://github.com/rvillamangca/AMA-ITE7107-DataSturctureAlgorithms>. Although the course is about C++ Data Structures implementation, it is noted that all exercises require Java language instead. Hence, the answers presented here are all written in Java.

Question 1: **Draw a similar picture of the tree when the List fields are implemented using linked lists.**



Answer:

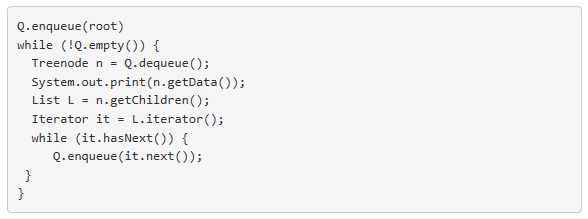
For a Linked List Tree implementation, the diagram would be almost the same except for the following:

* Since “items” will now point to a linked list instead of an array (as in the above diagram), “items” can have variable size.
* Instead of pointing to an empty array the “leaf” children’s “item” will now point to a “NULL” pointer.
* For efficiency, an additional field, to hold the address of the last item, will be required.

Considering the above, this student proposed the following Tree Diagram using Linked List Implementation:



Question 2: **Draw pictures of Q as it would be each time around the outer while loop in the code given below.**



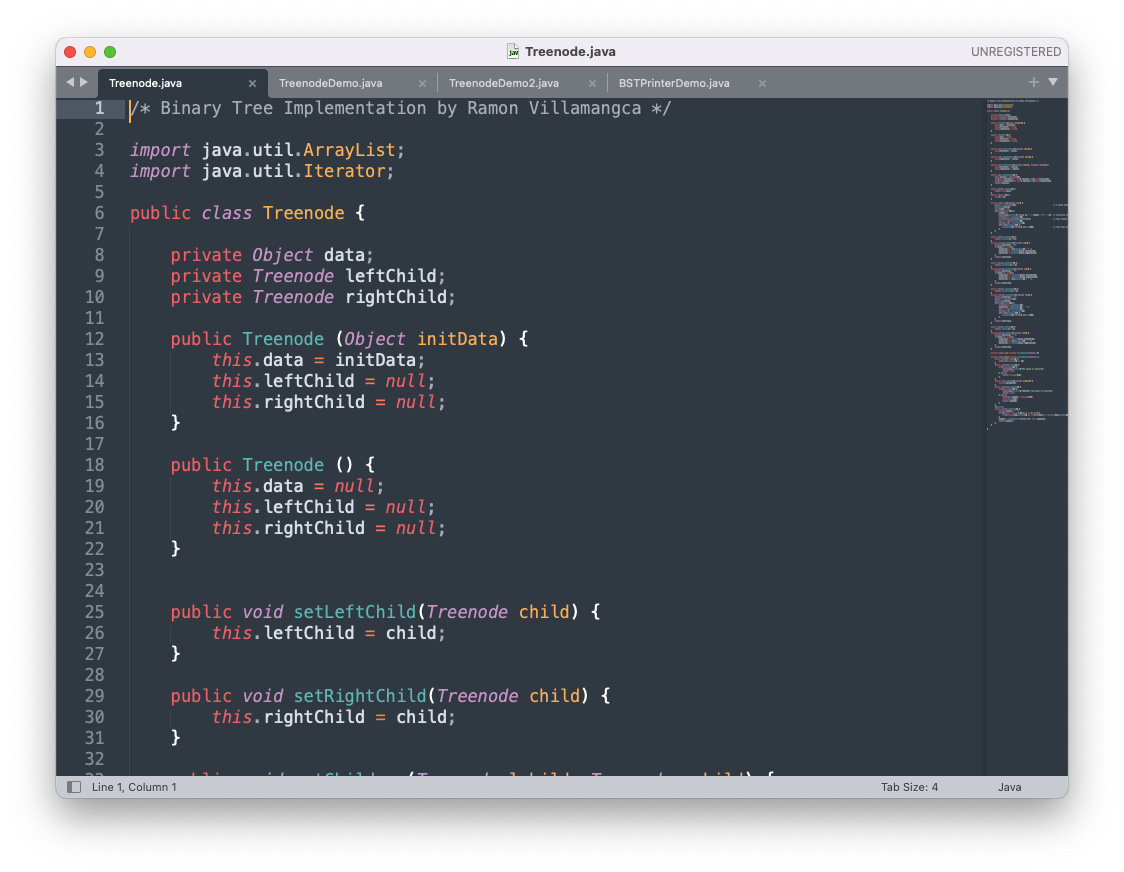
Answer:

By saying “pictures of Q”, we shall assume that this means: “print the contents of the queue Q”. However, even with this understanding, the code snippet listed above will NOT going to work because of the following:

* The tree to be traverse is unknown. With that, this student will consider the following tree structure for this exercise:

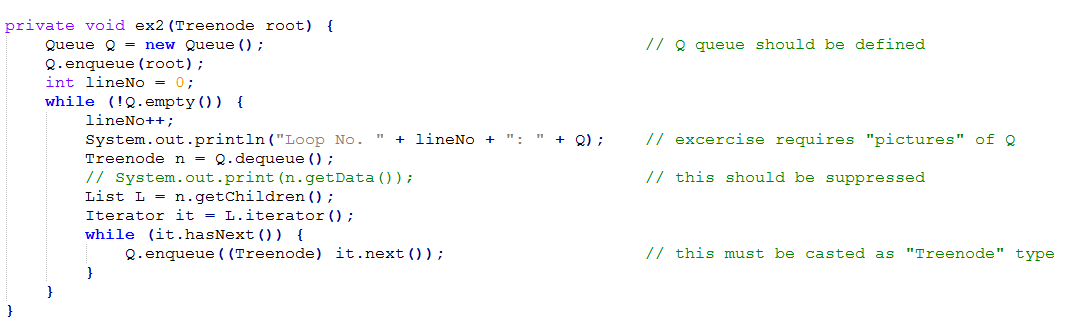


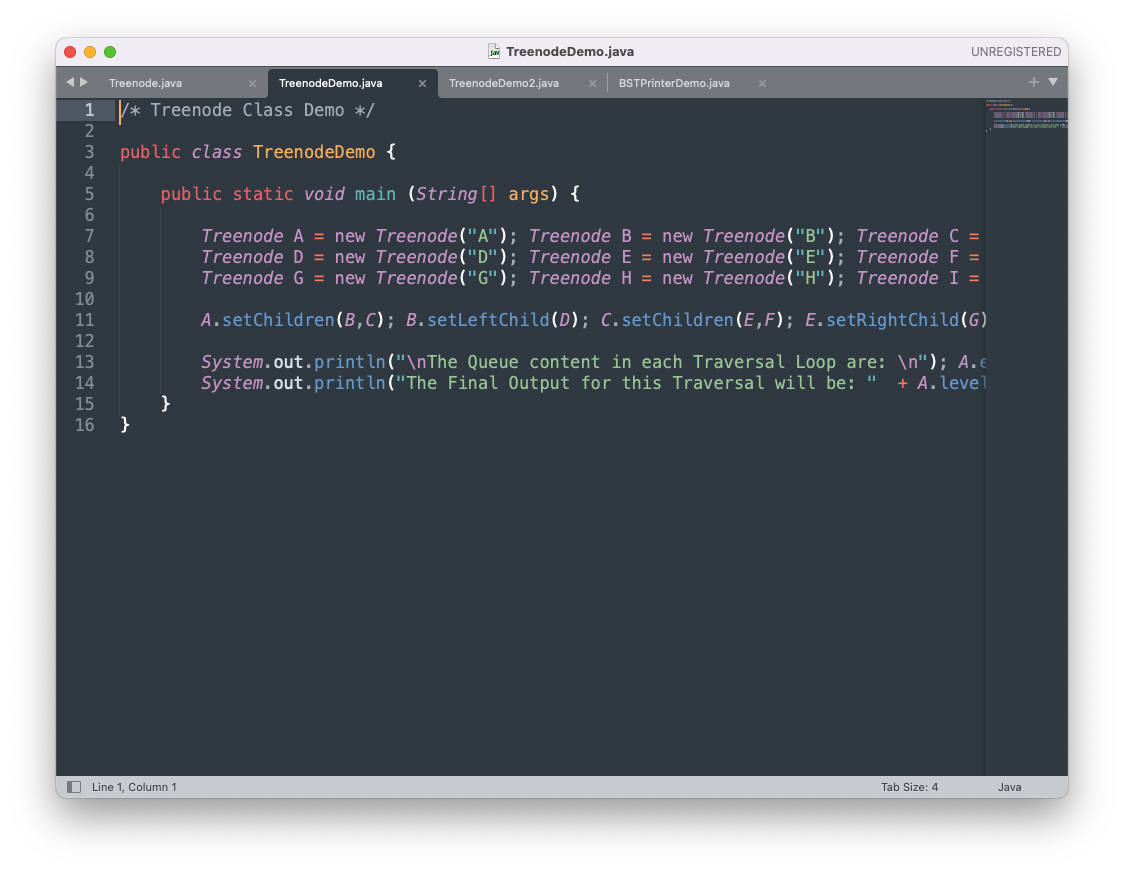
* The Tree Data Structure must be implemented first. To this, the following implementation of the “Treenode” class is proposed (see source code in the GitHub link provided above for the full program listing):

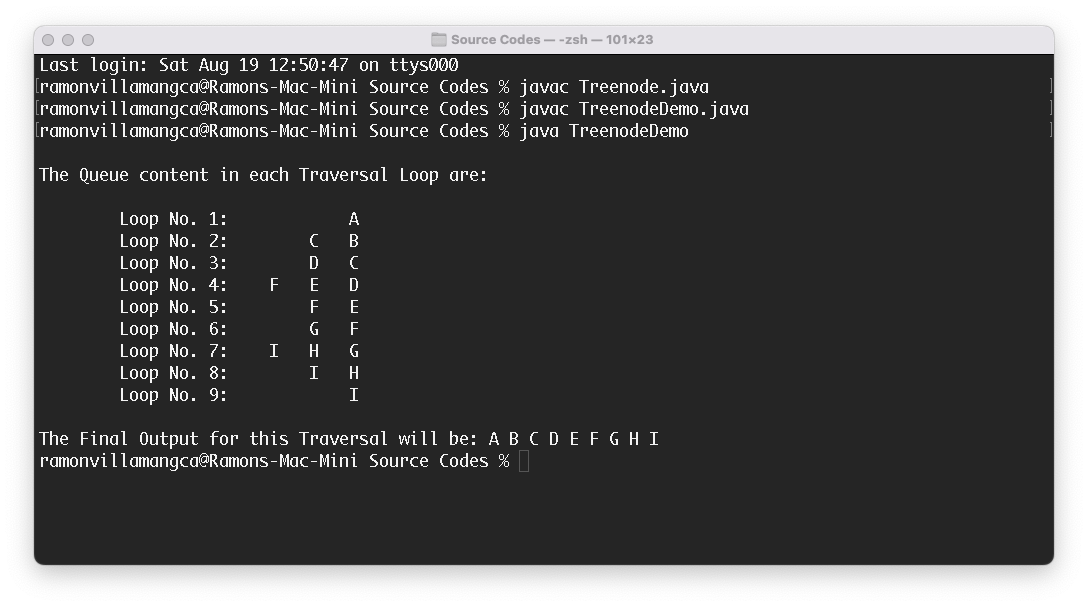


* The code snippet given is flawed. The following corrections are proposed:
  + Enclosed the code inside a function.
  + The code snippet as is will only output the final traversal result. A statement must be added to print the “Q” in each time the outer loop is invoked.

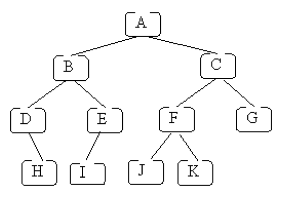
After the abovementioned considerations, below is the final code snippet, the driver (main) program code, followed by the screen output:







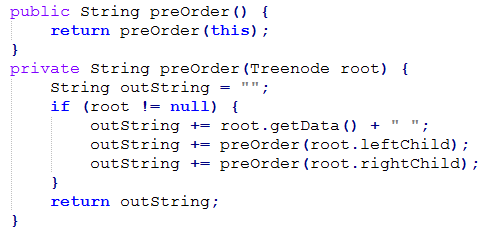
Question 3: **What is printed when the following tree is visited using (a) a preorder traversal, (b) a postorder traversal, (c) a level-order traversal, and (d) an in-order traversal?**



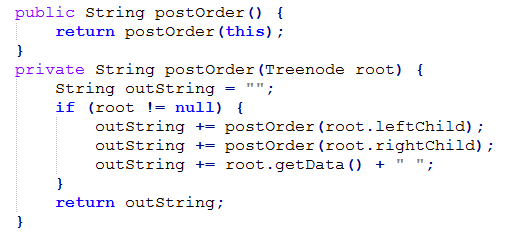
Answer:

Using the Tree implementation mentioned above, we added the following methods:

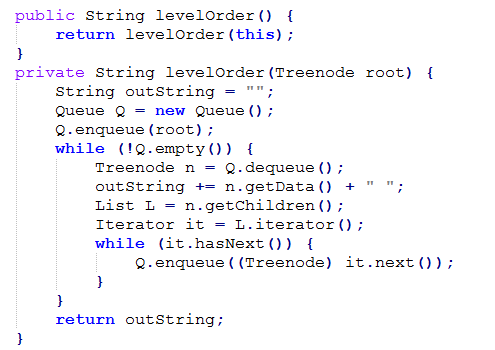
* Pre-order Traversal Method:



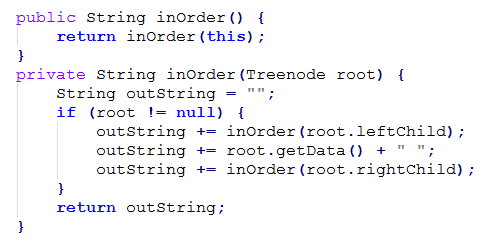
* Post-order Traversal Method:



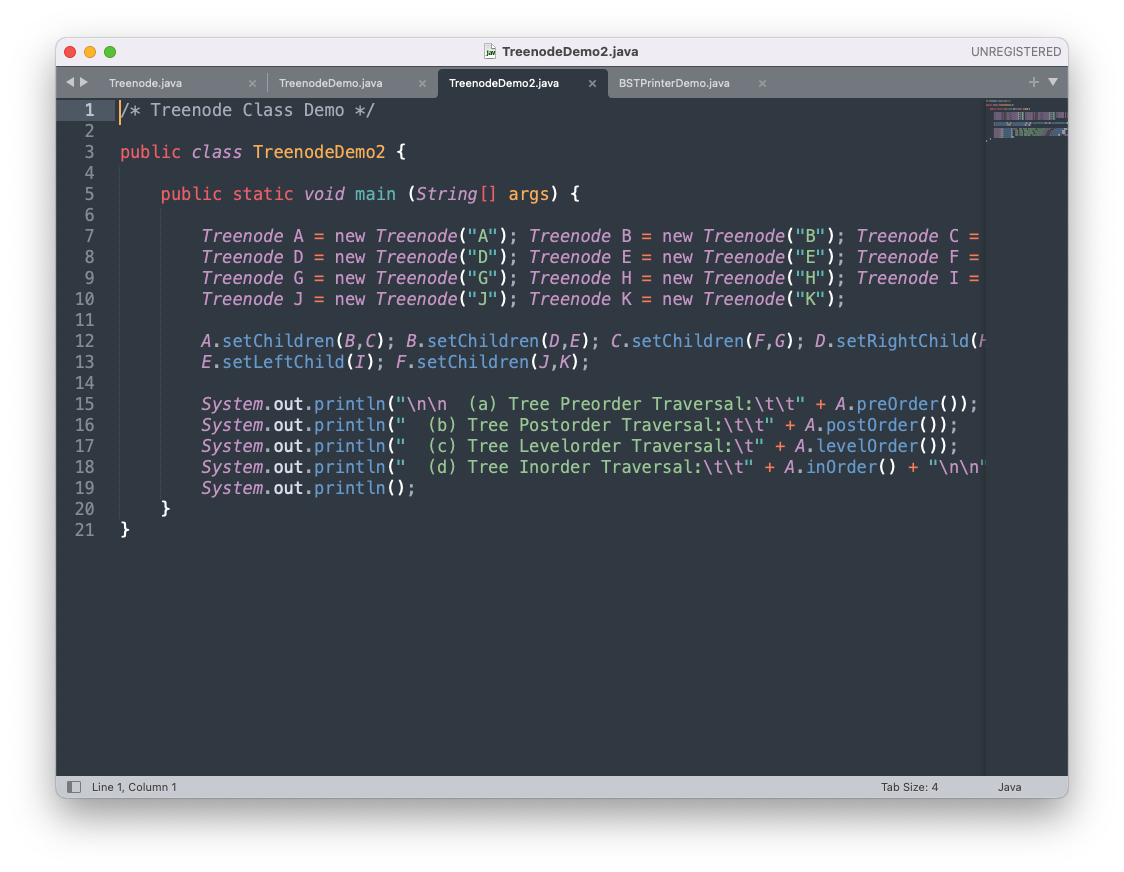
* Level-order Traversal Method:

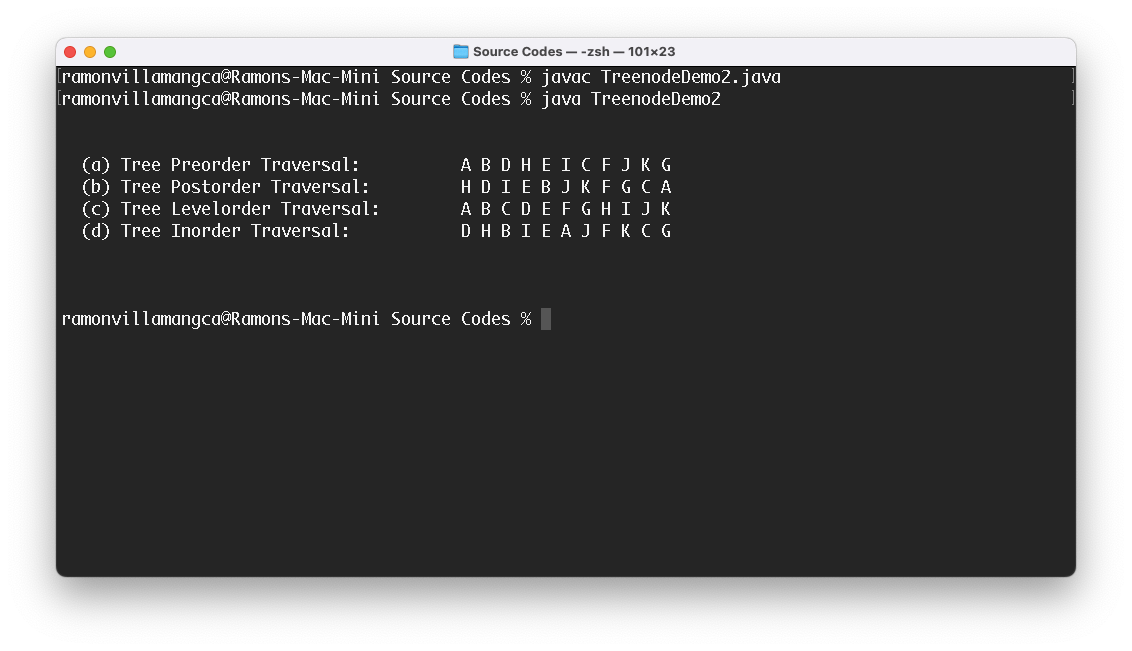


* In-order Traversal Method:



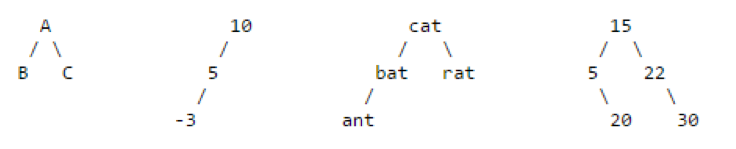
Applying the above methods to the given tree, yields the following result:





Question 4: **Question 4-1: Which of the following binary trees are BSTs? If a tree is not a BST, say why.**

**Question 4-2: Using which kind of traversal (preorder, postorder, inorder, or level-order) visits the nodes of a BST in sorted order?**



Answer:

**Answer 4-1:**  The first and the last tree are not BST.

When a child node is to be added in a Binary Search Tree (BST), this is attained by linking a child node with lesser value to the left of its parent and linking a child node with greater value to the right of its parent; traversing the tree (starting from the root) all the way down to the last leaf.

The reason for first tree, is that “B” is greater than “A” and therefore “B” should go to the right of “A”.

For the last tree, although 20 is greater than 5 (apparently should go to the right of 5), but the traversal should start from the root (15). This means that 20 should actually go to the right of 15 (to the left of 22, to be precise).

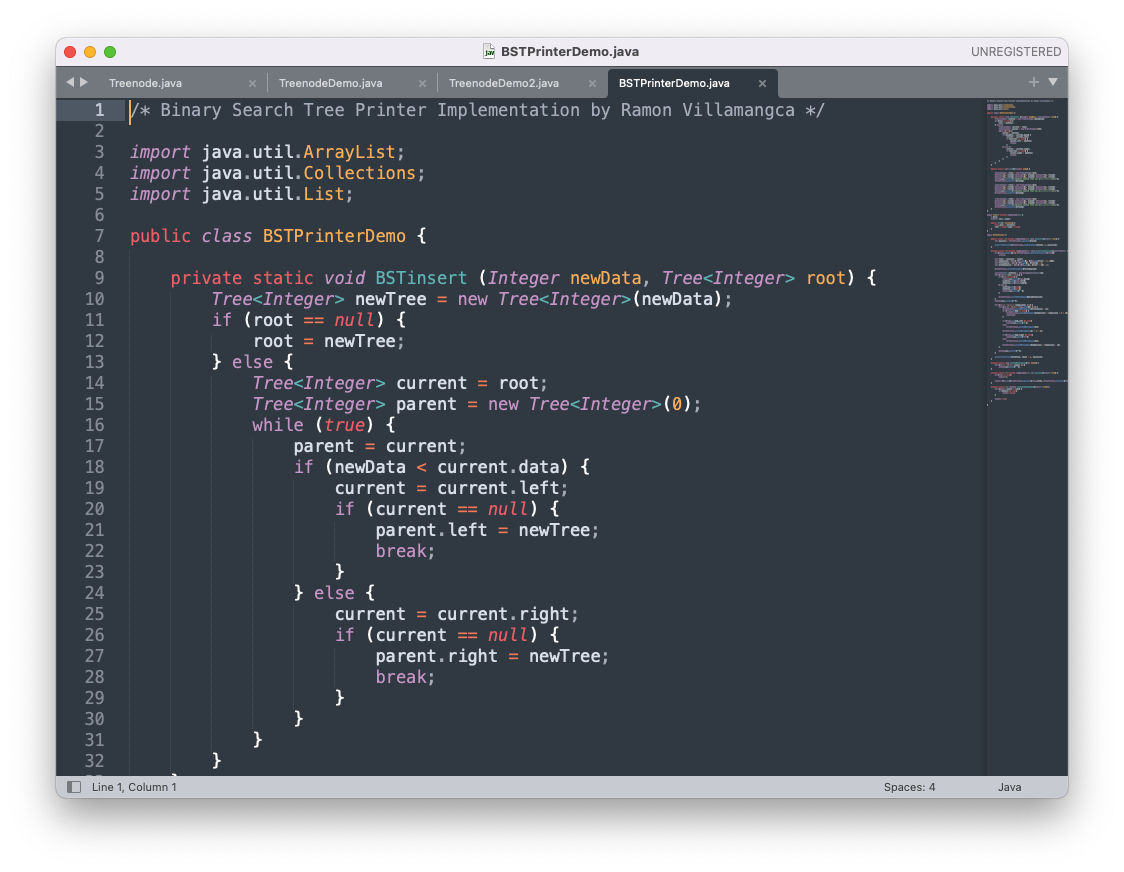
**Answer 4-2**  Under an In-order Traversal, BST nodes are visited in a sorted manner.

Question 5: **The order in which values are inserted determines what BST is built (inserting the same values in different orders can result in different final BSTs). Draw the BST that results from inserting the values 1 to 7 in each of the following orders (reading from left to right):**

* + 1. **5 3 7 6 2 1 4**
    2. **1 2 3 4 5 6 7**
    3. **4 3 5 2 6 1 7**

Answer:

For this exercise, a Binary Search Tree Printer (see the GitHub link provided above for the complete code listing), which draws the BST network on screen. The implementation is as follows:



With the following output results:

