# **Lab 6\_2 : Binary Search Tree Continued**

# **Week beginning 23rd October, 2023**

1. Use debug mode to step through the printSub() method to see the recursive calls.

2. Change the printSub() method to perform

* preorder traversal
* postorder traversal

Check that the output from these methods is what you expect.

3. Write a non-recursive version of inorder traversal. What data structure is required?

Use the following algorithm:

Create an empty stack

current = root

*// if the current node is null and the stack is also empty, we are done*

while ( stack is not empty OR current != null)

if (current != null)

*// if the current node exists, push it into the stack (defer it)  
 // and move to its left child*

push current onto stack

current = current.left

else

*// otherwise, if the current node is null, pop an element from  
 // the stack, visit it, and finally set the current node to it’s  
 // right child* pop the stack (save it to current) and visit it

current = current.right

To use a stack, use Deque interface as Java recommends. Use LinkedList as implementation of Deque.

4. Perform breadth-first traversal of a binary search tree. What data structure is required?

Use the following algorithm:

Start with empty data structure

insert the root (reference to root node) onto the data structure

while the data structure is not empty

remove a node from the data structure and visit it

insert its two child nodes //what order do you insert the children?