Assignment 03 – Part 01 (Unbalanced Binary Tree)

Write a program that implements a Binary Search Tree using a node-pointer implementation. Assume all entries will be unique.

```
class Node:
  def __init__(self, data=None):
      self.data = data
      self.left = None
      self.right = None
```

Requirements:

1) Implement the following functions:
   - void **insert** (int data) #creates a new Node and inserts it into correct location.
   - void **print_preorder**(Node root) #prints data in tree in pre-order traversal seq
   - void **print_postorder**(Node root) #prints data in tree in post-order traversal seq
   - void **print_inorder**(Node root) #prints data in tree in in-order traversal seq
   - int **search**(int data) #prints 0 if data item is not in tree; otherwise, returns number of nodes visited.
2) Insert the following sequence into the tree:

   Sequence A:
   1,5,4,6,7,2,3

   Sequence B:
   150,125,175,166,163,123,108,116,117,184,165,137,141,111,138,122,109,194,143,178,1
   73,139,126,170,190,140,188,120,195,113,104,193,181,185,198,103,182,136,115,191,14
   4,145,155,153,151,112,129,199,135,146,157,176,159,196,121,105,131,154,107,110,158
   ,187,134,132,179,133,102,172,106,177,171,156,168,161,149,124,189,167,174,147,148,
   197,160,130,164,152,142,162,118,186,169,127,114,192,180,101,119,128,100

3) Search **Sequence A** tree for values 1, 4, 2 and print the return values.
4) Search **Sequence B** tree for values 42, 142, 102, 190 and print the return values.
5) Traverse **Sequence A** tree with each of three traversal functions above and print the sequence of outputs for each of these traversals.
6) Traverse **Sequence B** tree with each of three traversal functions above and print the sequence of outputs for each of these traversals.
7) List and explain the complexities of each of the above algorithms from Part 01. This can be done within the comments at the end of the file or above each function header.