Practical No 7

Aim : Write A Program To Implement Binary Search Tree (BST)

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

class BST:
    def __init__(self):
        self.root = None

    def insert(self, key):
        """Insert a new node into the BST."""
        if self.root is None:
            self.root = Node(key)
        else:
            self._insert(self.root, key)

    def _insert(self, root, key):
        if key < root.key:
            if root.left is None:
                root.left = Node(key)
            else:
                self._insert(root.left, key)
        else:
            if root.right is None:
                root.right = Node(key)
            else:
                self._insert(root.right, key)

    # Traversals
    def inorder(self, root):
        """Left → Root → Right"""
        return self.inorder(root.left) + [root.key] + self.inorder(root.right) if root else []

    def preorder(self, root):
        """Root → Left → Right"""
        return [root.key] + self.preorder(root.left) + self.preorder(root.right) if root else []

    def postorder(self, root):
        """Left → Right → Root"""
        return self.postorder(root.left) + self.postorder(root.right) + [root.key] if root else []

# Example Usage
if __name__ == "__main__":
    # Dataset
    dataset = [50, 30, 20, 40, 70, 60, 80]
```

```
bst = BST()
for value in dataset:
    bst.insert(value)

print("Dataset:", dataset)
print("\n--- Tree Traversals ---")
print("In-order Traversal   :", bst.inorder(bst.root))
print("Pre-order Traversal  :", bst.preorder(bst.root))
print("Post-order Traversal :", bst.postorder(bst.root))

print("\nSorted sequence (from in-order):", bst.inorder(bst.root))
```

```
Dataset: [50, 30, 20, 40, 70, 60, 80]

--- Tree Traversals ---
In-order Traversal   : [20, 30, 40, 50, 60, 70, 80]
Pre-order Traversal  : [50, 30, 20, 40, 70, 60, 80]
Post-order Traversal : [20, 40, 30, 60, 80, 70, 50]

Sorted sequence (from in-order): [20, 30, 40, 50, 60, 70, 80]
```