****

**Riga Technical University**

**Telecommunications Software (RAE411).**

**Sixth Practical Exercise.**

**9th of May 2023**

**Zeyad Mohamed Nashaat Abdelghany.**

**230AMB013.**

**Spring 2023.**

1. **Introduction:**

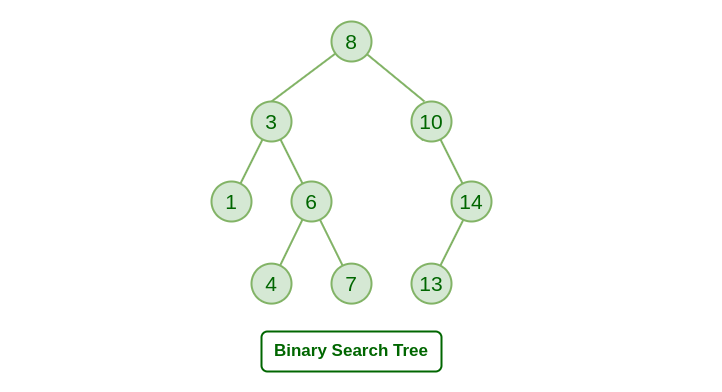
This report is the documentation for the sixth practical exercise. This exercise is divided into two sections: Binary search tree and SDN traffic classification with DT. Python is used to implement both sections using Anaconda.

1. **Binary Search Tree:**

Binary search tree is a node-based sorted data structure at which:

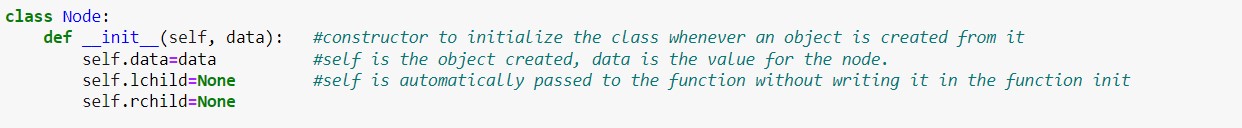
1. The left subtree of a node contains only nodes with lower keys’ values than the parents’ values.
2. The right subtree of a node contains only nodes with higher keys’ values than the parents’ values.
3. Each node has only a maximum value of two children.
4. There must be no duplicates for values in a tree.

Binary search trees are implemented to be used in the sorting algorithms. It is the basic data structure used in Microsoft Excel and Spreadsheets. Binary search trees have three main functions which are: searching, insertion, and deletion. There are also three ways to traverse through a binary tree which are: in-order, pre-order, and post-order traversing.



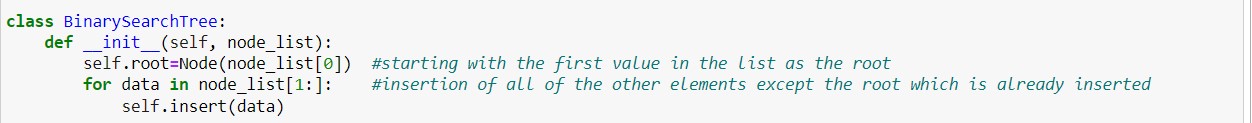
1. **Node creation:**

Nodes are the main building unit for a binary tree. A node has three properties: its value, its right child, and its left child. A class in python is done by the name of Node to create a node whenever it is needed in a tree. This class has a constructor which holds the data for the node, its left and right children.

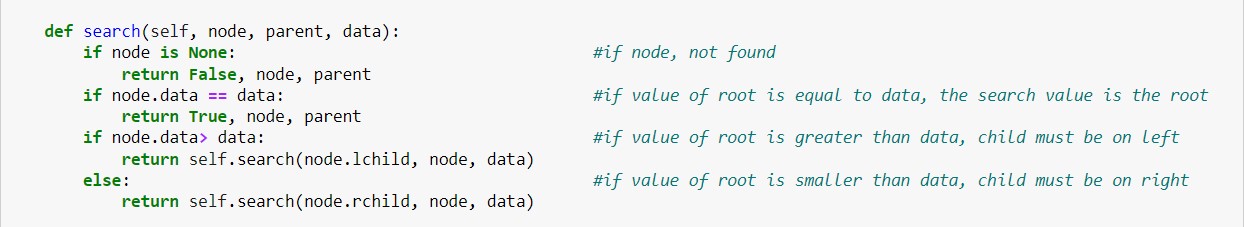


1. **Tree creation:**

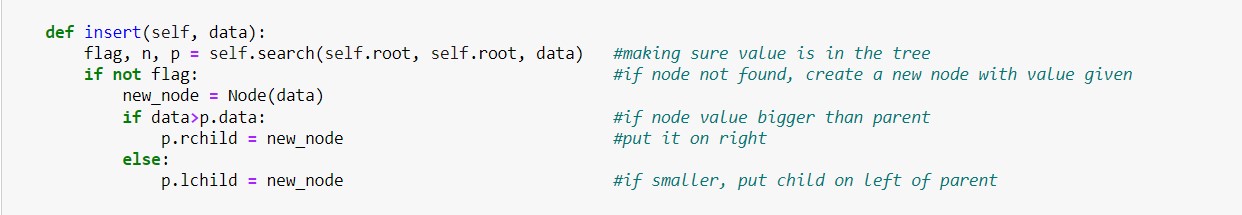
Binary tree is mainly multiple nodes connected to each other. A class is created in python to create a tree from a given list. The constructor of this class mainly takes the first element in the tree as the root node and insert the other elements in the with the insertion function following the rules of the binary tree data structure.



1. **Search function:**

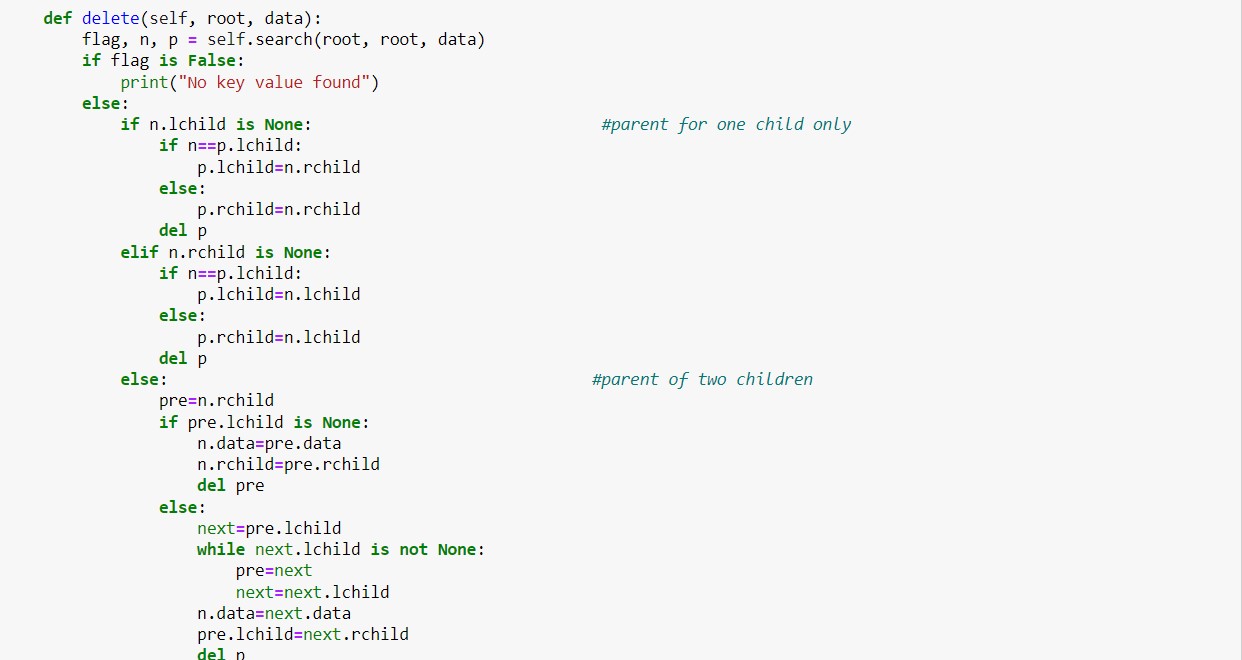
This function enables the user to insert a value for a node in the tree and return whether this value is found or not. It has 3 inputs: the tree root, the parent, and the value needed to be found. It is a recursive function that check whether the value is found in the tree or not.

1. **Insertion function:**

This function starts by making sure that the value inserted is not already in the tree to avoid duplicates. It is a recursive function that operates on the idea of creating a node to the value inserted. Then, the value of this node is then compared to its root. If the value is bigger than the root’s value, then the function recurs in the right direction. If it is smaller, then it recurs in the left direction.

1. **Deletion function:**

Deletion function has several scenarios and it depends whether the deleted node is a parent for one child or for two children. It makes sure at first that the needed node is present in the tree.

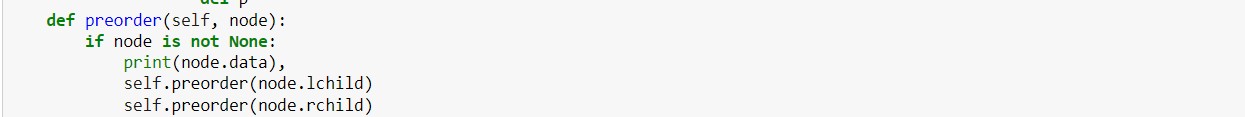


1. **Traversing functions:**

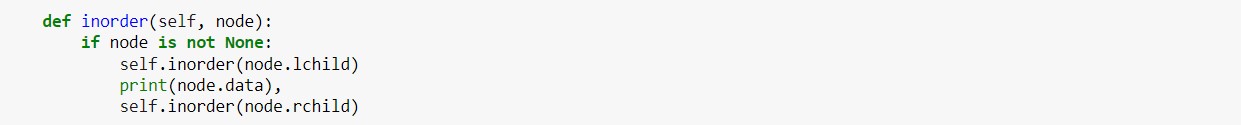
Traversing functions are used with the search trees to have an algorithm on how we extract the whole nodes in a tree. It is the process of visiting each node in the tree exactly once.

1. **Preorder traversing:**

This algorithm works by printing: the root, then printing the whole left subtree until reaching to the leaves, then traversing in the right subtree.

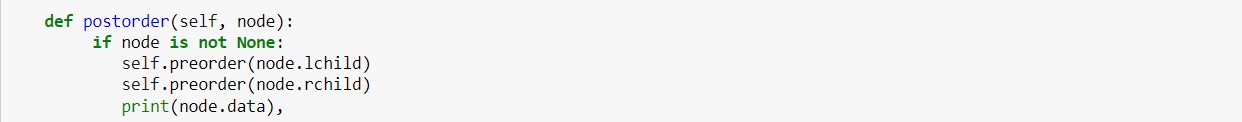
1. **Inorder traversing:**

This algorithm works by printing: the left node first, then the parent, then the right node.



1. **Postorder traversing:**

This algorithm works by printing: the left node, then the right node, then the parent.

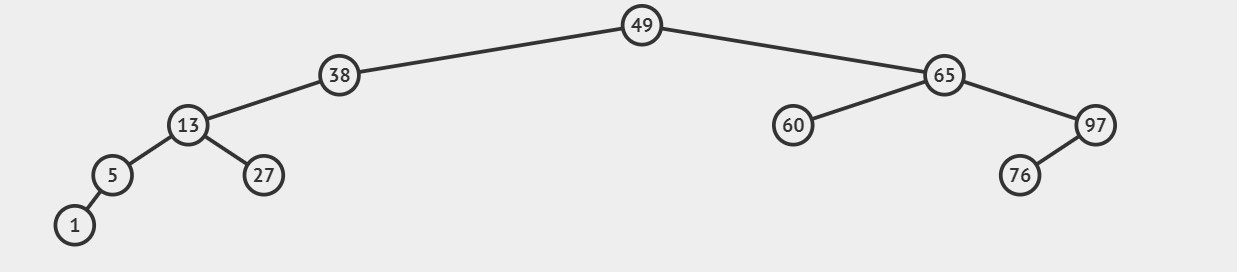
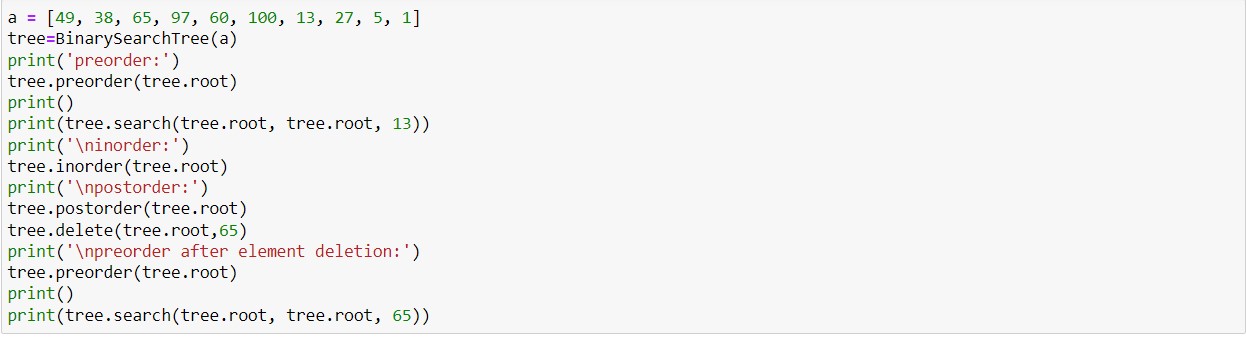


1. **Implementation on given lists:**

In this report, implementation of all of the three traversal method, deletion of element and searching for them is applied for each list.

1. **List a**:

List a is given as: [49, 38, 65, 97, 60, 76, 13, 27, 5, 1]

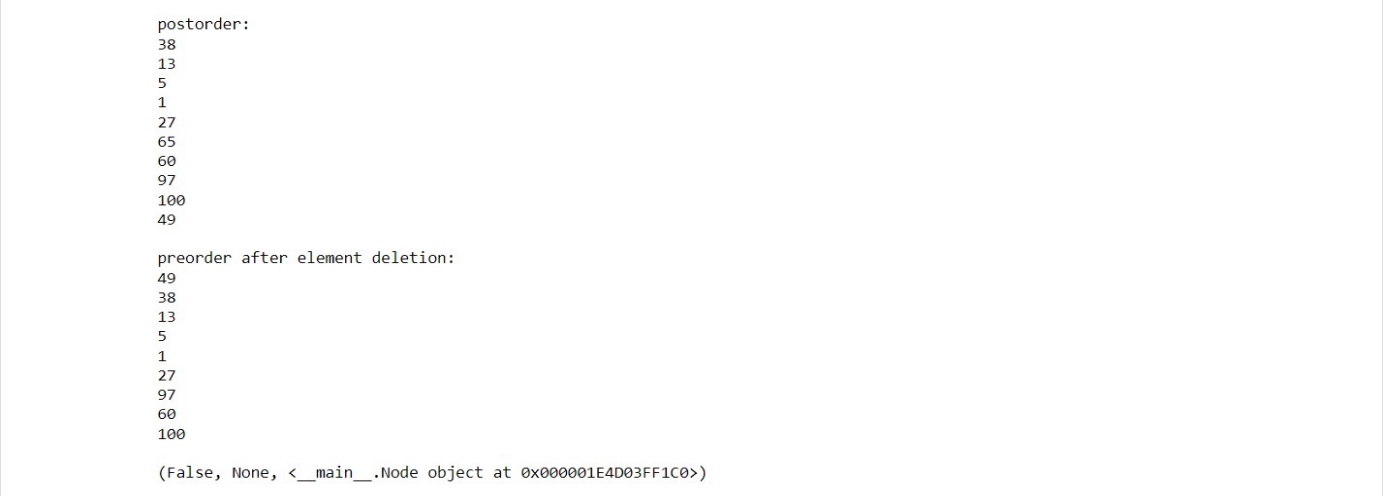


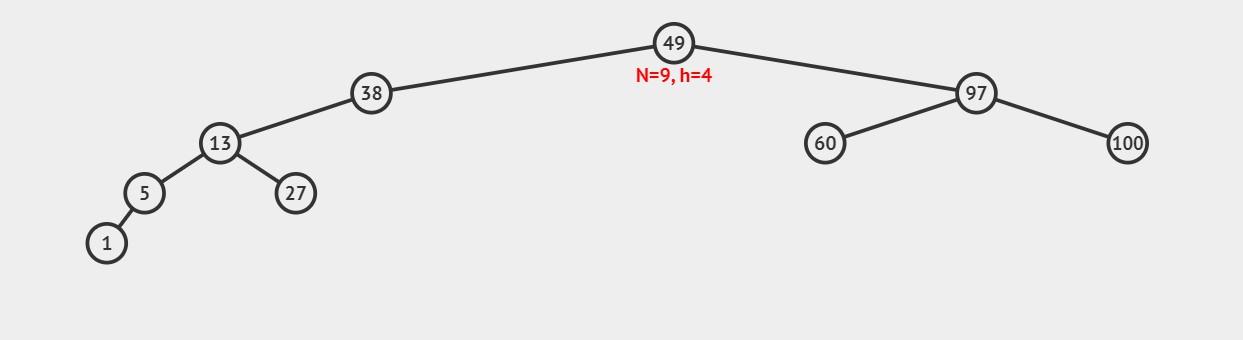
**Figure 2 Code for operations on List a**

**Figure 1 Original Tree for List a**

****

**Figure 3 Output for list a**

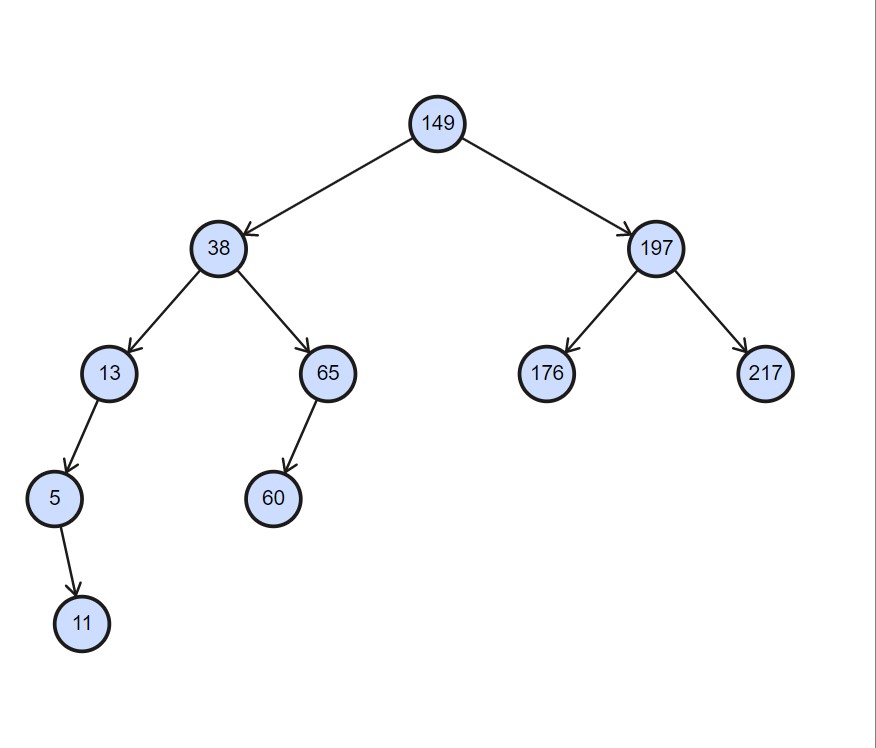




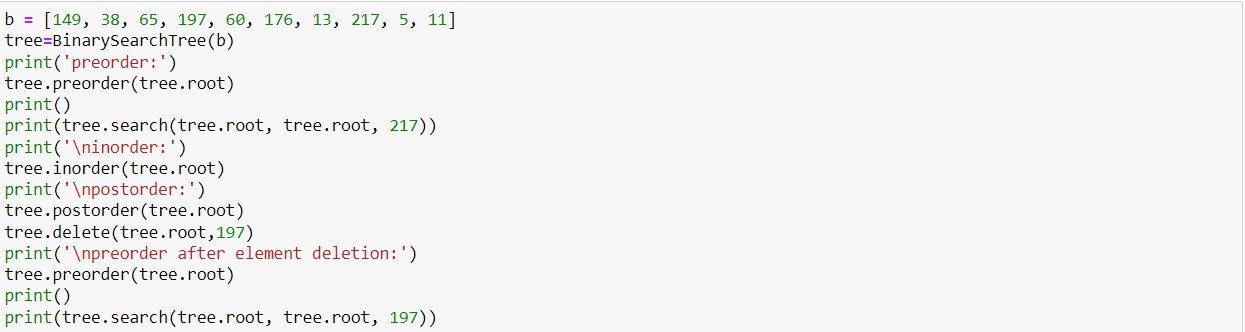
**Figure 4 Tree for list a after element deletion**

1. **List b:**

According to the given list: [149, 38, 65, 197, 60, 176, 13, 217, 5, 11].

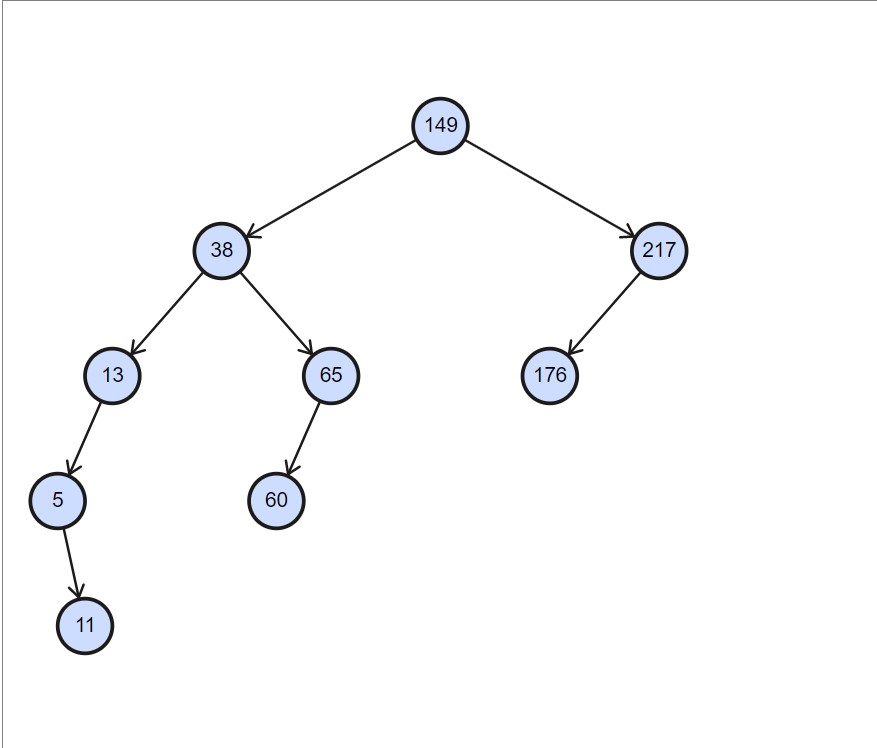


**Figure 5 Original Tree for List b**



**Figure 6 Code for operations on List b**



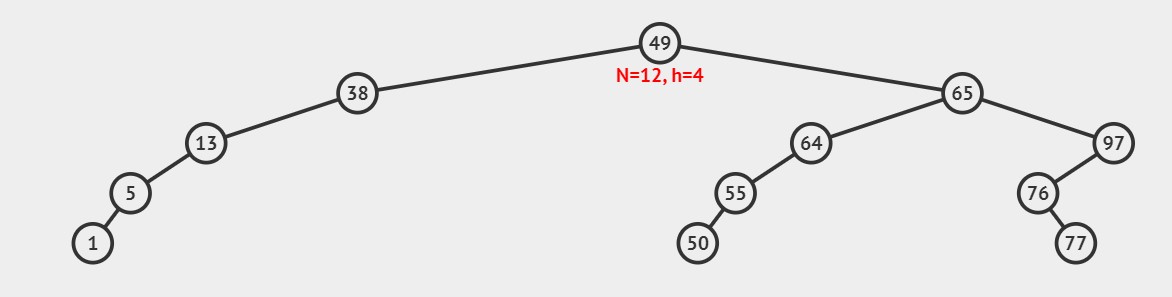


**Figure 8 Tree for list b after element deletion**

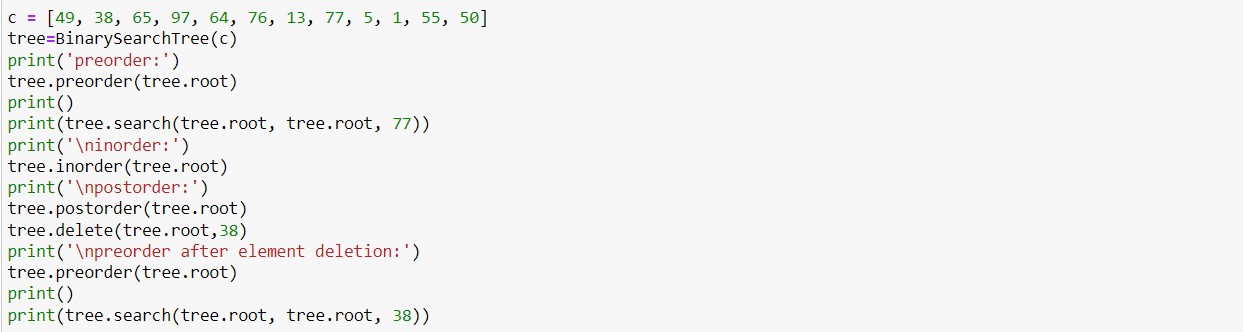
**Figure 7 Output for list b**

1. **List c:**

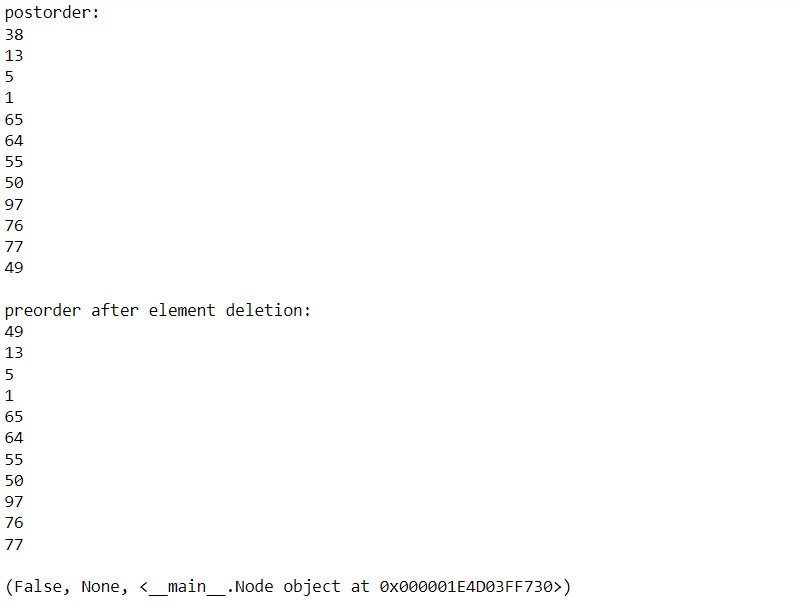
According to the given list: [49, 38, 65, 97, 64, 76, 13, 77, 5, 1, 55, 50].



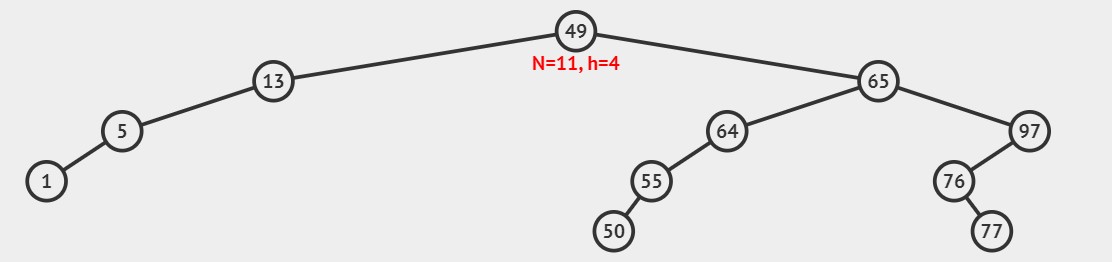
**Figure 9 Original Tree for List c**



**Figure 10 Code for operations on List c**



**Figure 11 Output for list c**

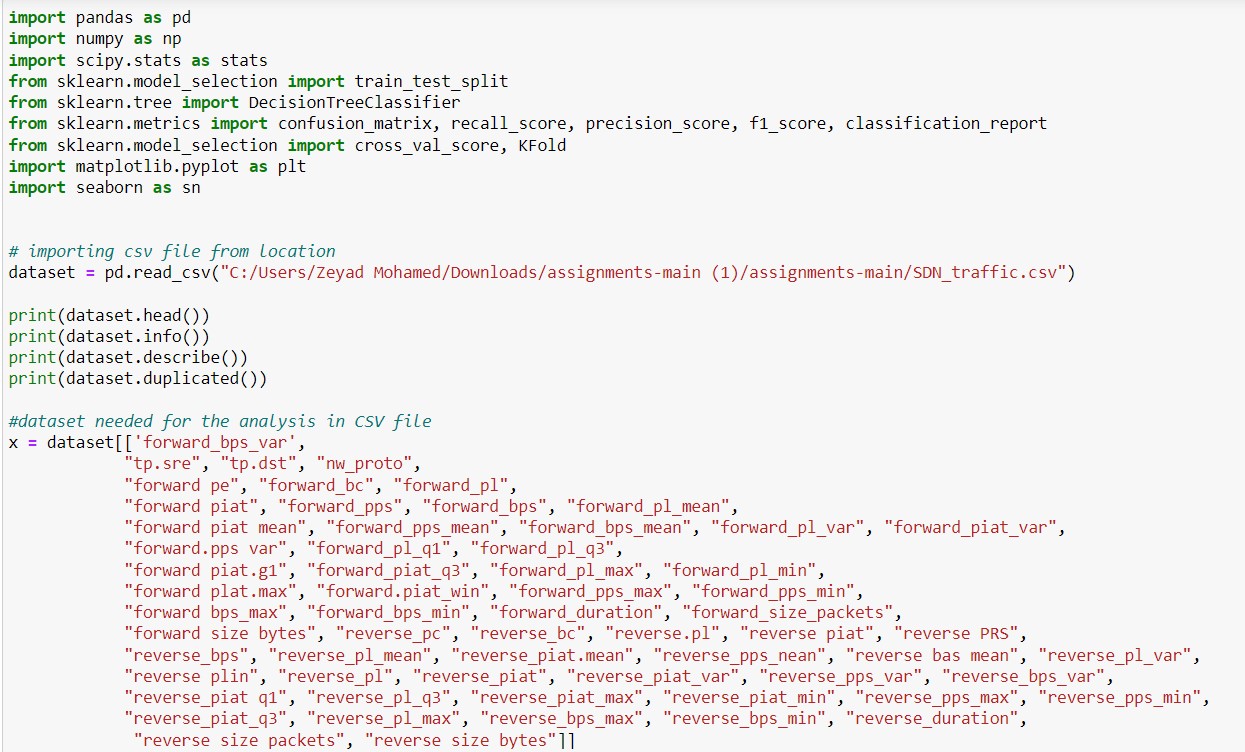


**Figure 12 Tree for list c after element deletion**

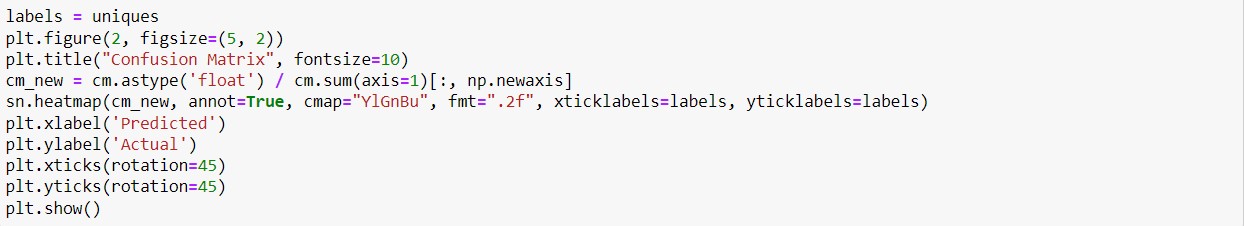
1. **SDN classification with decision tree:**

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

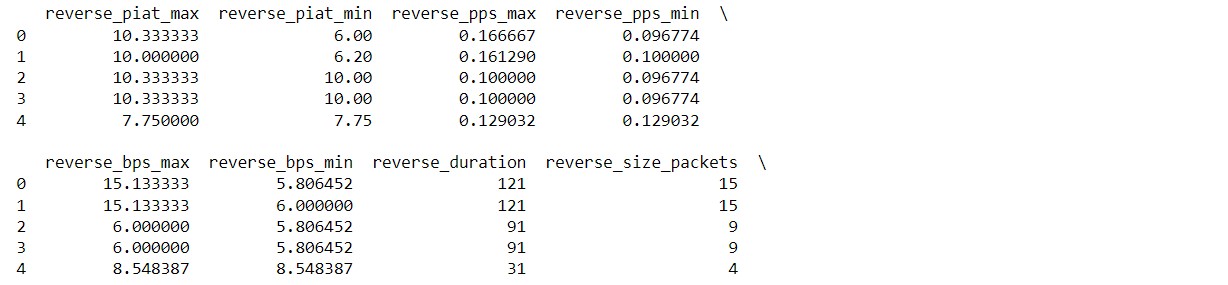
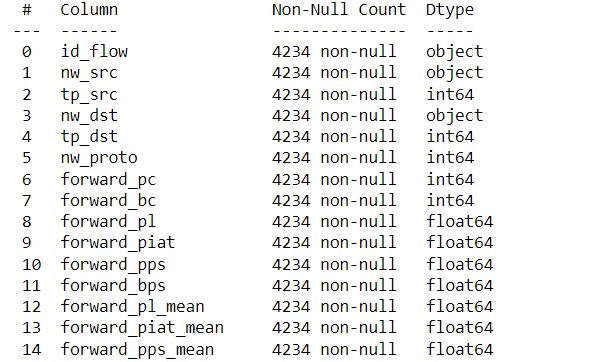
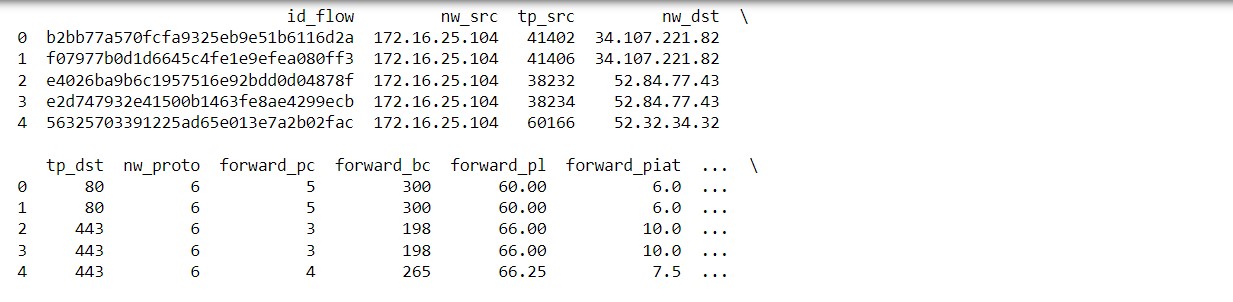
1. **Program Code:**

This code is used to perform some data analysis and classification tasks using a decision tree classifier on a dataset stored in a CSV file.





1. **Code Output:**

****

