Data Structures and Algorithms II - Assignment 01

Name: K.K.G.H.H. Priyangika

Index: AS2016480

- **delete() Method**

```
public boolean  delete(int key) {
        Node current = root;
        Node parent = root;
        boolean isLeftChild = true;

    // searching node
    while(current.iData != key) {
        parent = current;
        if (key < current.iData) {
            isLeftChild = true;
            current = current.leftChild;
        } else {
            isLeftChild = false;
            current = current.rightChild;
                }
        if(current == null)
            return false ;
      }
    // case 1 : no children
    if (current.leftChild == null && current.rightChild == null) {
            if (current == root)
                    root = null;
            else if (isLeftChild)
                    parent.leftChild = null;
             else
                    parent.rightChild = null;
     }
```

```
// case 2 : one child
// case 2(i) : no right child
else if (current.rightChild == null)
        if (current == root)
                root = current.leftChild ;
        else if (isLeftChild)
                parent.leftChild = current.leftChild;
         else
                parent.rightChild = current.leftChild ;

// case 2(ii) : no left child
else if(current.leftChild == null)
       if (current == root)
              root = current.rightChild ;
       else if(isLeftChild)
              parent.leftChild = current.rightChild ;
       else
              parent.rightChild = current.rightChild ;

// case 3 : two children ( find & replace with successor)
else
{
         // get successor of delNode
         Node successor =  getSuccessor (current);
         if (current == root)
                root = successor;
          else if (isLeftChild)
                parent.leftChild = successor;
```

```
                else
                    parent.rightChild = successor;

                //get current's left child connection
                successor.leftChild = current.leftChild;
            }
        return true ;
    }
```

- **getSuccessor() method**

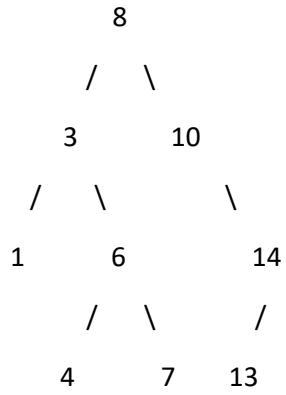
```
    private Node getSuccessor (Node delNode) {
        Node tempParent = null ;
        Node temp = delNode.rightChild ;

        while (temp.leftChild != null){
            tempParent = temp;
            temp = temp.leftChild;
        }

        if (temp != delNode.rightChild){
            tempParent.leftChild = temp.rightChild;
        }

      return temp;
    }
```

Binary tree ;



The NetBeans out put is :

Inorder traversal of the given tree

1 3 4 6 7 8 10 13 14

Delete 1 -> (no child)

Inorder traversal of the modified tree

3 4 6 7 8 10 13 14

Delete 14 -> (one child)

Inorder traversal of the modified tree

3 4 6 7 8 10 13

Delete 6 -> (two children)

Inorder traversal of the modified tree

3 4 7 8 10 13

BUILD SUCCESSFUL (total time: 0 seconds)