TreeType\* deleteBST(TreeType\* root, int key)

{

    TreeType \* = root;

    TreeType \* parent = NULL;

1. //if the tree is empty

If(root==NULL)

Print”Underflow..”

   // Check if the key is actually present in the BST. The pointer parent points to the parent of the node to be deleted.

    while ( != NULL && ->data != key) {

        parent = ;

        if (key < ->data)

             = ->left;

        else

             = ->right;

    }

1. Unsuccessful search

    if ( == NULL) {

        Print “The element doesn’t exist”;

        return root;

    }

1. Deletion of node with one child

    // Check if the node to be deleted has atmost one child.

    if (temp->left == NULL || temp->right == NULL) {

                  TreeType \* newCurr;

        // if the left child does not exist.

        if (temp ->left == NULL)

            newCurr = temp->right;

        else

            newCurr = temp->left;

        // check if the node to be deleted is the root.

        if (parent == NULL)

            return newCurr;

        // check if the node to be deleted is parent's left or right child

        // and then replace this with newCurr

        if (curr == parent->left)

            parent->left = newCurr;

        else

            parent->right = newCurr;

    }

    // node to be deleted has

    // two children.

    else {

        treeNode\* p = NULL;

        treeNode\* ;

        // Compute the inorder successor

        Temp1 = curr->right;

        while (temp1->left != NULL) {

            p = temp1 ;

           temp1  = temp1->left;

        }

        // check if the parent of the inorder

        // successor is the root or not.

        // if it isn't, then make the

        // left child of its parent equal to the

        // inorder successor's right child.

        if (p != NULL)

            p->left = ->right;

        // if the inorder successor was the

        // root, then make the right child

        // of the node to be deleted equal

        // to the right child of the inorder

        // successor.

        else

            curr->right = ->right;

        curr->data = ->data;

        free();

    }

    return root;

}