PRACTICAL

1.Implement binary search tree.

Code:

```
#include <stdio.h>
#include <stdlib.h>

struct node {
int key;
 struct node *left, *right;
};

// Create a node
struct node *newNode(int item) {
 struct node *temp = (struct node *)malloc(sizeof(struct node));
temp->key = item; temp->left = temp->right = NULL; return
temp;
}

// Inorder Traversal void
inorder(struct node *root) { if
```

```
 (root != NULL) { // Traverse left
inorder(root->left);

 // Traverse root
printf("%d -> ", root->key);

 // Traverse right inorder(root-
>right);
 }
}

// Insert a node
struct node *insert(struct node *node, int key) {
// Return a new node if the tree is empty if
(node == NULL) return newNode(key);

 // Traverse to the right place and insert the node
if (key < node->key) node->left = insert(node-
>left, key); else
 node->right = insert(node->right, key);


  return node;
```

```
}

// Find the inorder successor
struct node *minValueNode(struct node *node) {
struct node *current = node;

 // Find the leftmost leaf while (current
&& current->left != NULL) current =
current->left;

 return current;
}

// Deleting a node
struct node *deleteNode(struct node *root, int key) {
// Return if the tree is empty
if (root == NULL) return root;

 // Find the node to be deleted
if (key < root->key)

root->left = deleteNode(root->left, key);
else if (key > root->key)
```

```
root->right = deleteNode(root->right, key);

else {
// If the node is with only one child or no child
if (root->left == NULL) { struct node *temp =
root->right; free(root); return temp;
} else if (root->right == NULL) {
struct node *temp = root->left;
free(root); return temp;
}

// If the node has two children
struct node *temp = minValueNode(root->right);

// Place the inorder successor in position of the node to be deleted
root->key = temp->key;

 // Delete the inorder successor
root->right = deleteNode(root->right, temp->key);   }



return root;
```

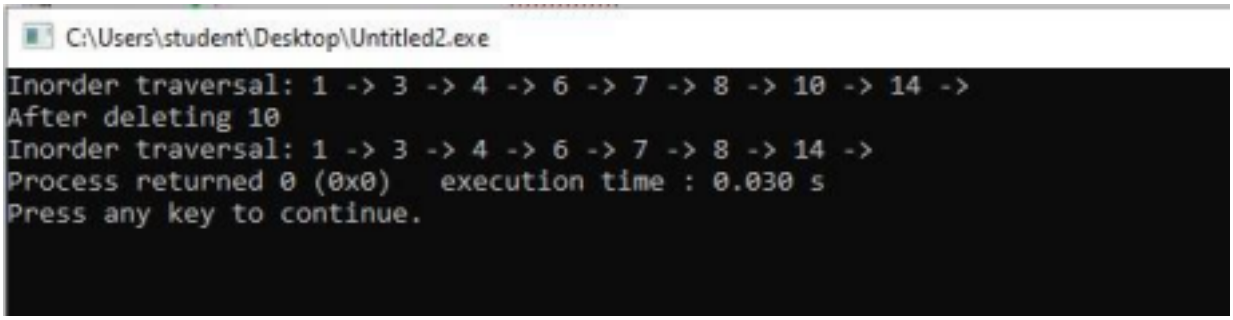
```
}

// Driver code int main() {
struct node *root = NULL;
root = insert(root, 8);
root = insert(root, 3);
root = insert(root, 1);
root = insert(root, 6);
root = insert(root, 7);
root = insert(root, 10);
root = insert(root, 14);
root = insert(root, 4);

 printf("Inorder traversal: ");
inorder(root);

 printf("\nAfter deleting 10\n");
root = deleteNode(root, 10);
printf("Inorder traversal: ");
inorder(root);
}
```

Output:


C:\Users\student\Desktop\Untitled2.exe
Inorder traversal: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 14 ->
After deleting 10
Inorder traversal: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 14 ->
Process returned 0 (0x0)   execution time : 0.030 s
Press any key to continue.