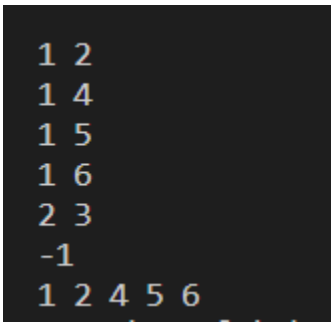
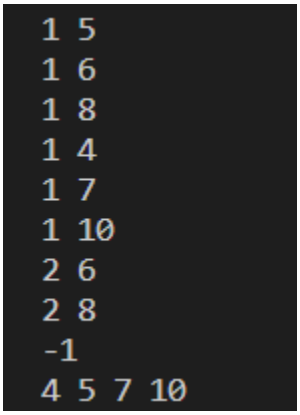
Lab 07

Test case 01:



Test case 02:



Code:

```
#include <iostream>
using namespace std;

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

// Inorder traversal
void traverseInOrder(struct node *root) {
  if (root ==NULL){
    return;
  }
  traverseInOrder(root->left);
  cout<< root->key<<" ";
  traverseInOrder(root ->right);
}

// Insert a node
struct node* insertNode(struct node* node, int key) {
    if (node == NULL) {
        node = new struct node();
        node->key = key;
        node->left = NULL;
        node->right = NULL;
        return node;
    }
    if (key > node->key) {
        node->right = insertNode(node->right, key);
    } else if (key <= node->key) {
        node->left = insertNode(node->left, key);
    }
    return node;
}


// Deleting a node
struct node *deleteNode(struct node *root, int key) {
  if(root == NULL){
    return root;
  }else if(key<root->key){
    root->left = deleteNode(root->left,key);
  }else if(key>root->key){
```

```
    root->right = deleteNode(root->right,key);
  }else{
    //no children
    if(root->left==NULL && root->right == NULL){
      delete root;
      root = NULL;
    }//1 child
    else if(root->left==NULL){
      struct node* temp = root->right;
      delete root;
      return temp;
    }else if(root->right==NULL){
      struct node* temp = root->left;
      delete root;
      return temp;
    }//2 children
    else{
      struct node* temp = root->right;
        while (temp->left != NULL) {
            temp = temp->left;
        }
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
  }
  return root;
}

// Driver code
int main() {
  struct node *root = NULL;

  int operation;
  int operand;
  cin >> operation;

  while (operation != -1) {
    switch(operation) {
      case 1: // insert
        cin >> operand;
        root = insertNode(root, operand);
        cin >> operation;
        break;
      case 2: // delete
        cin >> operand;
```

```
      root = deleteNode(root, operand);
      cin >> operation;
      break;
    default:
      cout << "Invalid Operator!\n";
      return 0;
  }
 }

 traverseInOrder(root);
}
```

GitHub:

https://github.com/Thejas0604/CSE-labs/tree/main/Lab%2007