**NAME:** ALEENA SAEED

**SAP ID:** 56270

**DSA LAB 13 TASKS**

**TASK 1:**

#include<iostream>

using namespace std;

class Node

{

public:

int data;

Node \*left;

Node \*right;

Node(int value)

{

data=value;

left=right=nullptr;

}

};

class BST

{

private:

Node \*root;

Node \*insert(Node \*node,int value)

{

if(node==nullptr)

{

return new Node(value);

}

if(value<node->data)

{

node->left=insert(node->left,value);

}

else if(value>node->data)

{

node->right=insert(node->right,value);

}

return node;

}

void inorder(Node \*node)

{

if(node!=nullptr)

{

inorder(node->left);

cout<<node->data<<" ";

inorder(node->right);

}

}

void preorder(Node \*node)

{

if(node!=nullptr)

{

cout<<node->data<<" ";

preorder(node->left);

preorder(node->right);

}

}

void postorder(Node \*node)

{

if(node!=nullptr)

{

postorder(node->left);

postorder(node->right);

cout<<node->data<<" ";

}

}

public:

BST()

{

root=nullptr;

}

void insert(int value)

{

root=insert(root,value);

}

void inorder()

{

cout<<"Inorder Traversal:";

inorder(root);

cout<<endl;

}

void preorder()

{

cout<<"Preorder Traversal:";

preorder(root);

cout<<endl;

}

void postorder()

{

cout<<"Postorder Traversal:";

postorder(root);

cout<<endl;

}

};

int main()

{

BST tree;

int values[]={12,7,9,10,22,24,30,18,3,14,20};

for(int value:values)

{

tree.insert(value);

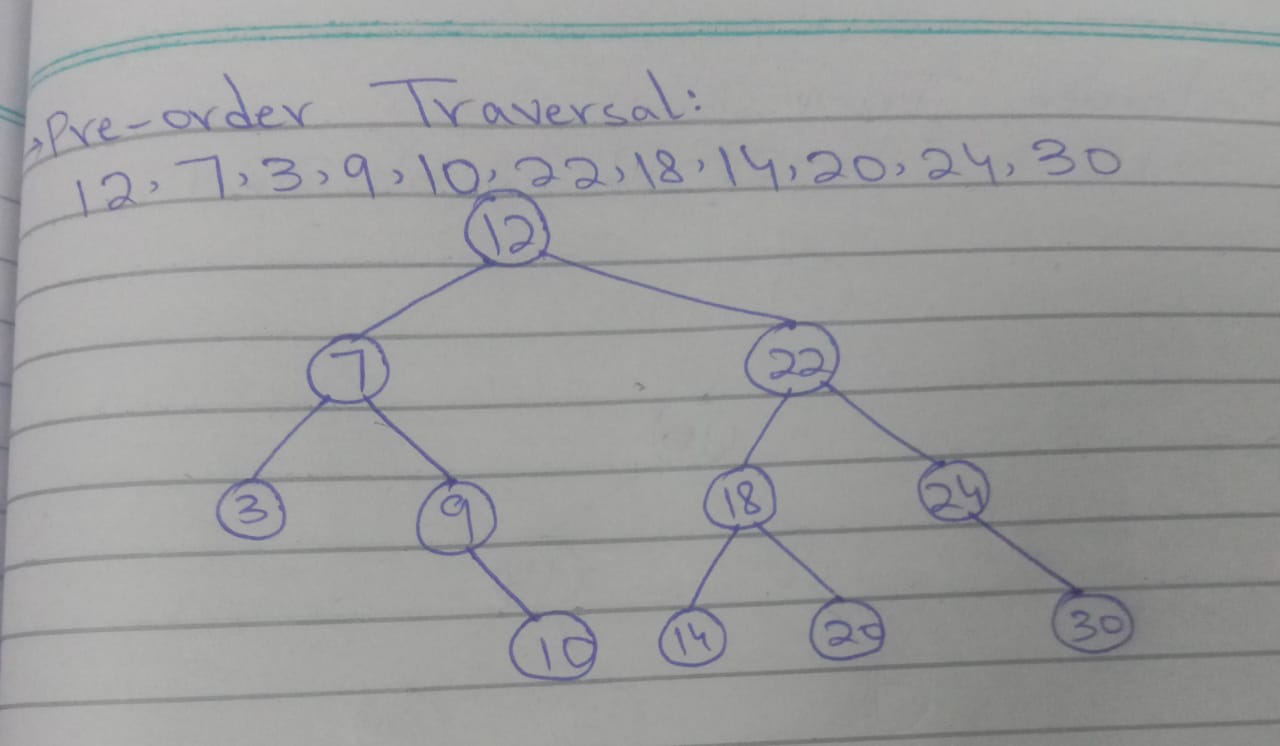
}

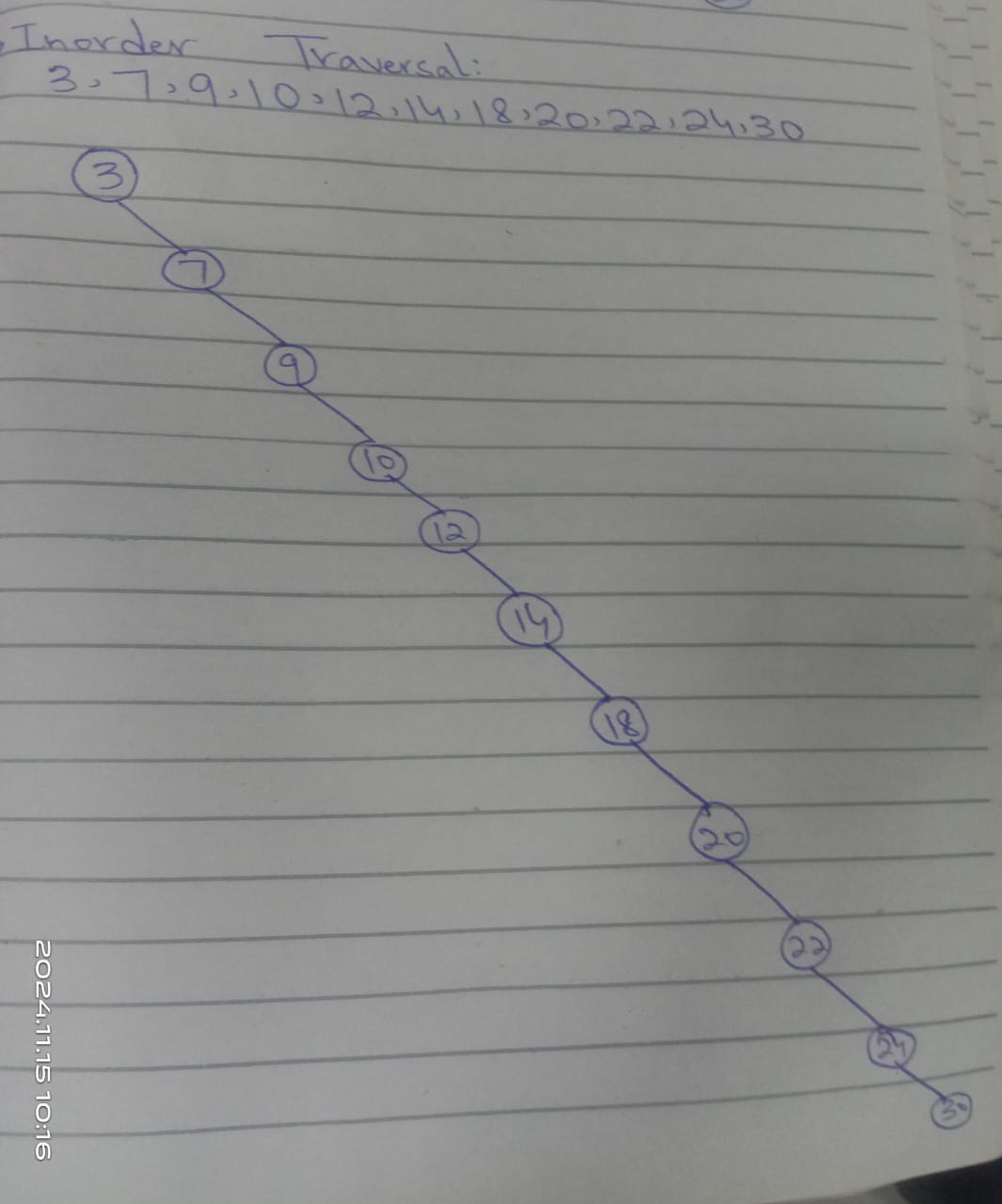
tree.inorder();

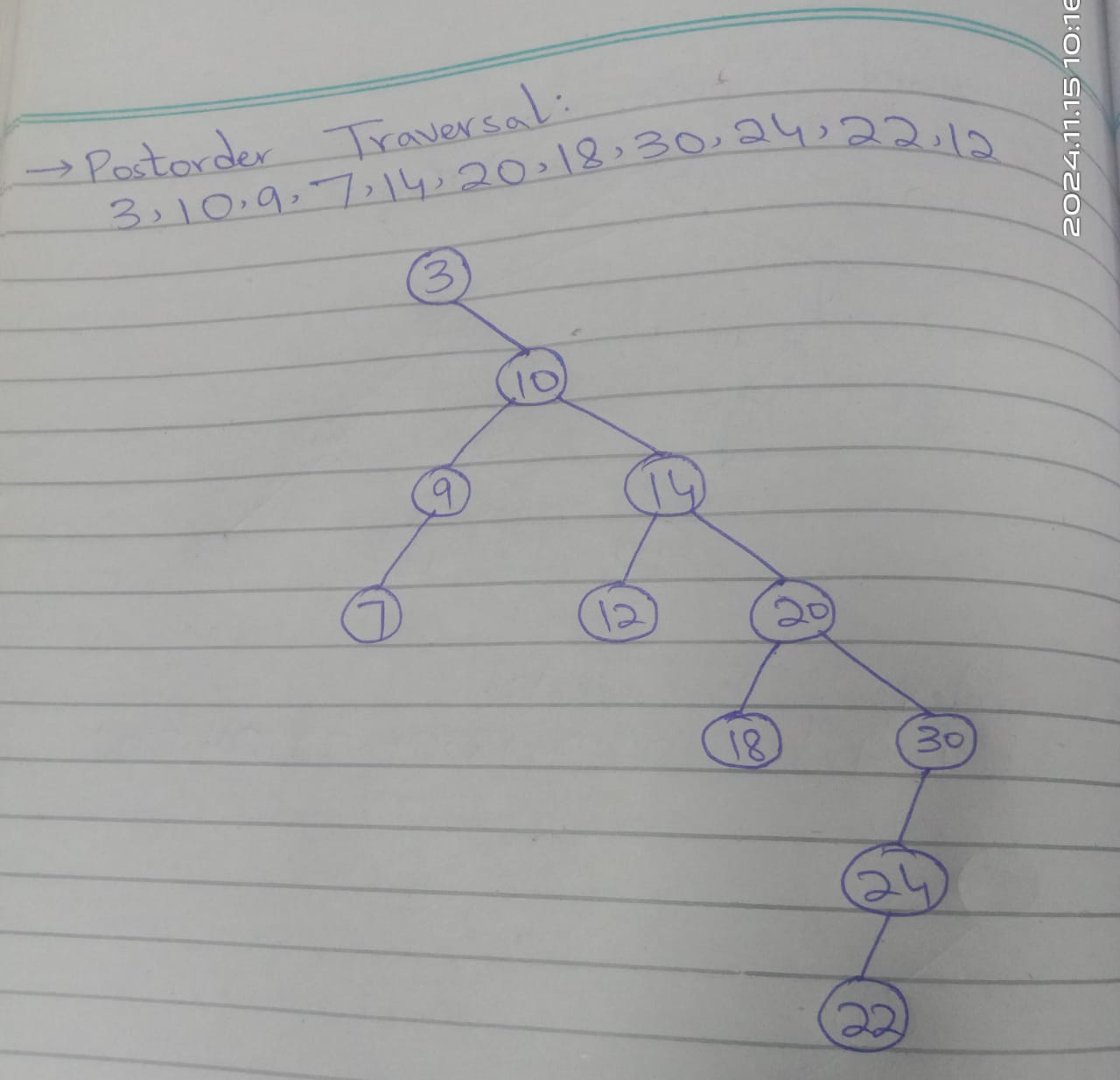
tree.preorder();

tree.postorder();

return 0;

}



**TASK 2:**

#include <iostream>

using namespace std;

struct Employee

{

int empNo;

string name;

double salary;

Employee \*left;

Employee \*right;

Employee(int no,string n,double sal)

{

empNo=no;

name=n;

salary=sal;

left=nullptr;

right=nullptr;

}

};

class BST

{

private:

Employee \*root;

Employee \*insert(Employee \*node,int empNo,string name,double salary)

{

if(node==nullptr)

return new Employee(empNo,name,salary);

if(empNo<node->empNo)

node->left=insert(node->left,empNo,name,salary);

else if(empNo>node->empNo)

node->right=insert(node->right,empNo,name,salary);

return node;

}

Employee \*search(Employee \*node,int empNo)

{

if(node==nullptr || node->empNo==empNo)

return node;

if(empNo<node->empNo)

return search(node->left,empNo);

return search(node->right,empNo);

}

Employee \*findMin(Employee \*node)

{

while(node->left!=nullptr)

node=node->left;

return node;

}

Employee \*deleteNode(Employee \*node,int empNo)

{

if(node==nullptr)

return node;

if(empNo<node->empNo)

node->left=deleteNode(node->left,empNo);

else if(empNo>node->empNo)

node->right=deleteNode(node->right,empNo);

else

{

if(node->left==nullptr)

{

Employee \*temp=node->right;

delete node;

return temp;

}

else if(node->right==nullptr)

{

Employee \*temp=node->left;

delete node;

return temp;

}

Employee \*temp=findMin(node->right);

node->empNo=temp->empNo;

node->name=temp->name;

node->salary=temp->salary;

node->right=deleteNode(node->right,temp->empNo);

}

return node;

}

void inOrder(Employee \*node)

{

if(node!=nullptr)

{

inOrder(node->left);

cout<<"EmpNo:"<<node->empNo<<",Name:"<<node->name<<",Salary:"<<node->salary<<endl;

inOrder(node->right);

}

}

void preOrder(Employee \*node)

{

if(node!=nullptr)

{

cout<<"EmpNo:"<<node->empNo<<",Name:"<<node->name<<",Salary:"<<node->salary<<endl;

preOrder(node->left);

preOrder(node->right);

}

}

void postOrder(Employee \*node)

{

if(node!=nullptr)

{

postOrder(node->left);

postOrder(node->right);

cout <<"EmpNo:"<<node->empNo<<",Name:"<<node->name<<",Salary:"<<node->salary<<endl;

}

}

public:

BST()

{

root=nullptr;

}

void insertEmployee(int empNo,string name,double salary)

{

root=insert(root,empNo,name,salary);

}

void searchEmployee(int empNo)

{

Employee \*result=search(root,empNo);

if(result != nullptr)

cout<<"Employee Found: EmpNo:"<<result->empNo<<", Name: " << result->name << ", Salary: " << result->salary << endl;

else

cout << "Employee Not Found!" << endl;

}

void deleteEmployee(int empNo) {

root = deleteNode(root, empNo);

cout << "Employee with EmpNo " << empNo << " deleted successfully (if existed)." << endl;

}

void displayInOrder() {

cout << "In-Order Traversal:\n";

inOrder(root);

}

void displayPreOrder() {

cout << "Pre-Order Traversal:\n";

preOrder(root);

}

void displayPostOrder() {

cout << "Post-Order Traversal:\n";

postOrder(root);

}

};

int main() {

BST tree;

tree.insertEmployee(101, "Alice", 50000.0);

tree.insertEmployee(102, "Bob", 60000.0);

tree.insertEmployee(100, "Charlie", 45000.0);

tree.displayInOrder();

tree.displayPreOrder();

tree.displayPostOrder();

tree.searchEmployee(102);

tree.deleteEmployee(101);

tree.displayInOrder();

return 0;

}