#include <iostream>

using namespace std;

class node {

public:

    int Id;

    string department;

    node \*left, \*right;

    node() {

        Id = 0;

        left = NULL;

        right = NULL;

    }

    node(int data, string dept) {

        Id = data;

        department = dept;

        left = NULL;

        right = NULL;

    }

    int getvalue() {

        return Id;

    }

    node\* getright() {

        return right;

    }

    node\* getleft() {

        return left;

    }

};

class BST {

public:

    node \*root, \*n1;

    BST() {

        root = NULL;

    }

    node\* checkifNodeExist(node \*rootnode, int id) {

        if (rootnode == NULL || rootnode->getvalue() == id)

            return rootnode;

        if (id > rootnode->getvalue())

            return checkifNodeExist(rootnode->getright(), id);

        else

            return checkifNodeExist(rootnode->getleft(), id);

    }

    void insertNode(int data, string dept) {

        n1 = new node(data, dept);

        if (root == NULL) {

            root = n1;

            cout << "Node added as root node\n";

        }

        else {

            if (checkifNodeExist(root, data) == NULL) {

                node \*temp = root;

                while (true) {

                    int val = temp->getvalue();

                    if (data > val) {

                        if (temp->getright() == NULL) {

                            temp->right = n1;

                            cout << "Node added to the right\n";

                            break;

                        }

                        else {

                            temp = temp->getright();

                        }

                    }

                    else if (data < val) {

                        if (temp->getleft() == NULL) {

                            temp->left = n1;

                            cout << "Node added to the left\n";

                            break;

                        }

                        else {

                            temp = temp->getleft();

                        }

                    }

                    else {

                        cout << "Node already Exists with the same value\n";

                        break;

                    }

                }

            }

            else {

                cout << "Node already Exists with same value\n";

            }

        }

    }

    void findSubordinatesRecursive(node \*root, int Id) {

        if (root == nullptr)

            return;

        if (root->getvalue() == Id) {

            cout << "Subordinates of Employee " << root->department << " (ID: " << root->getvalue() << "):\n";

            if (root->getleft())

                cout << " - " << root->getleft()->department << " (ID: " << root->getleft()->getvalue() << ")\n";

            if (root->getright())

                cout << " - " << root->getright()->department << " (ID: " << root->getright()->getvalue() << ")\n";

            return;

        }

        findSubordinatesRecursive(root->getleft(), Id);

        findSubordinatesRecursive(root->getright(), Id);

    }

    node\* searchforEmp(node \*root, int Id) {

        if (root == nullptr || root->getvalue() == Id)

            return root;

        if (Id < root->getvalue())

            return searchforEmp(root->getleft(), Id);

        return searchforEmp(root->getright(), Id);

    }

    void viewAllEmployeeDetailsRecursive(node \*root) {

        if (root == nullptr)

            return;

        viewAllEmployeeDetailsRecursive(root->getleft());

        cout << "Employee ID: " << root->getvalue() << ", department: " << root->department << endl;

        viewAllEmployeeDetailsRecursive(root->getright());

    }

};

int main() {

    BST employeeTree;

    int choice;

    do {

        cout << "\nEmployee Management System Menu:\n";

        cout << "1. Insert an employee\n";

        cout << "2. Search for an employee\n";

        cout << "3. Find subordinates of an employee\n";

        cout << "4. View all employee details\n";

        cout << "5. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

        {

            int empID;

            string department;

            cout << "Enter Employee ID: ";

            cin >> empID;

            cout << "Enter Employee Name: ";

            cin.ignore();

            cin >> department;

            employeeTree.insertNode(empID, department);

            break;

        }

        case 2:

        {

            int empID;

            cout << "Enter Employee ID to search: ";

            cin >> empID;

            node \*emp = employeeTree.searchforEmp(employeeTree.root, empID);

            if (emp != nullptr)

                cout << "Employee Found - department: " << emp->department << endl;

            else

                cout << "Employee not found.\n";

            break;

        }

        case 3:

        {

            int empID;

            cout << "Enter Employee ID to find subordinates: ";

            cin >> empID;

            employeeTree.findSubordinatesRecursive(employeeTree.root, empID);

            break;

        }

        case 4:

        {

            employeeTree.viewAllEmployeeDetailsRecursive(employeeTree.root);

            break;

        }

        case 5:

        {

            cout << "Exiting...\n";

            break;

        }

        default:

            cout << "Invalid choice. Please try again.\n";

        }

    } while (choice != 5);

    return 0;

}

OUTPUTS:







