**Aimen nadeem**

**2022-BSE-01**

**3A**

**DATA STRUCTURES**

**PROJECT**

**BANK MANAGEMENT SYSTEM**

**Statement:**

The provided C++ program represents a simple Bank Management System utilizing a binary search tree (BST) to organize and manage bank accounts. The program is within a Bank class, which includes a Account structure. Each account is characterized by an account number (accNum), an account holder name (accHolderNm), a balance, and pointers to left and right child nodes for the binary search tree structure.

**PROGRAM:**

#include<iostream>

using namespace std;

class Bank {

struct Account {

int accNum;

char accHolderNm[50];

double balance;

Account\* leftchild;

Account\* rightchild;

} \* root;

public:

Bank() : root(NULL) {}

void addAccount() {

Account\* newAccount = new Account;

newAccount->leftchild = NULL;

newAccount->rightchild = NULL;

cout << "Enter account number: ";

cin >> newAccount->accNum;

cin.ignore();

cout << "Enter account holder name: ";

cin.getline(newAccount->accHolderNm, 50);

cout << "Enter initial balance: ";

cin >> newAccount->balance;

root = insert(root, newAccount);

}

Account\* insert(Account\* node, Account\* newAccount) {

if (node == NULL)

return newAccount;

if (newAccount->accNum < node->accNum)

node->leftchild= insert(node->leftchild, newAccount);

else if (newAccount->accNum > node->accNum)

node->rightchild = insert(node->rightchild, newAccount);

return node;

}

void deleteAccount() {

int accNum;

cout << "Enter account number to delete: ";

cin >> accNum;

root = deleteNode(root, accNum);

}

Account\* deleteNode(Account\* root, int key) {

if (root == NULL)

return root;

if (key < root->accNum)

root->leftchild= deleteNode(root->leftchild, key);

else if (key > root->accNum)

root->rightchild = deleteNode(root->rightchild, key);

else {

if (root->leftchild == NULL) {

Account\* temp = root->rightchild;

delete root;

return temp;

}

else if (root->rightchild == NULL) {

Account\* temp = root->leftchild;

delete root;

return temp;

}

Account\* temp = minValueNode(root->rightchild);

root->accNum = temp->accNum;

root->rightchild = deleteNode(root->rightchild, temp->accNum);

}

return root;

}

Account\* minValueNode(Account\* node) {

Account\* current = node;

while (current->leftchild!= NULL)

current = current->leftchild;

return current;

}

void displayAccounts() {

if (root == NULL)

cout << "No accounts.\n";

else

displayTree(root);

}

void displayTree(Account\* node) {

if (node != NULL) {

displayTree(node->leftchild);

cout << "Account Number: " << node->accNum << endl;

cout << "Account Holder Name: " << node->accHolderNm << endl;

cout << "Balance: $" << node->balance << endl;

cout << endl;

displayTree(node->rightchild);

}

}

};

int main() {

Bank b;

char ch;

int choice;

do {

cout << "-----------------------------\n";

cout << "BANK MANAGEMENT SYSTEM\n";

cout << "-----------------------------\n";

cout << "1) Add Account\n";

cout << "2) Delete Account\n";

cout << "3) Display Accounts\n";

cout << "4) Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1:

b.addAccount();

break;

case 2:

b.deleteAccount();

break;

case 3:

b.displayAccounts();

break;

}

if (choice != 4) {

cout << "Press y to continue: ";

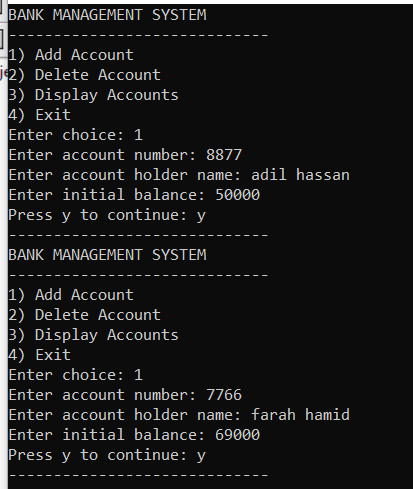
cin >> ch;

}

} while (choice != 4);

return 0;

}

**Output:**

