#include <iostream>

#include <vector>

#include <string>

Using namespace std;

// Structure to represent an account

Struct Account {

String account\_number;

Float balance;

};

// Function to create an account

Void createAccount(vector<Account> &accounts) {

Account newAccount;

Cout << “Enter account number: “;

Cin >> newAccount.account\_number;

Cout << “Enter initial balance: “;

Cin >> newAccount.balance;

Accounts.push\_back(newAccount);

Cout << “Account created successfully.\n”;

}

// Function to perform withdrawal

Void withdrawal(vector<Account> &accounts) {

String accountNumber;

Float amount;

Cout << “Enter account number: “;

Cin >> accountNumber;

For (auto &account : accounts) {

If (account.account\_number == accountNumber) {

Cout << “Enter withdrawal amount: “;

Cin >> amount;

If (amount > account.balance) {

Cout << “Insufficient funds.\n”;

} else {

Account.balance -= amount;

Cout << “Withdrawal successful. Remaining balance: “ << account.balance << “\n”;

}

Return;

}

}

Cout << “Account not found.\n”;

}

// Function to perform deposit

Void deposit(vector<Account> &accounts) {

String accountNumber;

Float amount;

Cout << “Enter account number: “;

Cin >> accountNumber;

For (auto &account : accounts) {

If (account.account\_number == accountNumber) {

Cout << “Enter deposit amount: “;

Cin >> amount;

Account.balance += amount;

Cout << “Deposit successful. New balance: “ << account.balance << “\n”;

Return;

}

}

Cout << “Account not found.\n”;

}

// Function to perform payment

Void payment(vector<Account> &accounts) {

String senderAccount, receiverAccount;

Float amount;

Cout << “Enter sender account number: “;

Cin >> senderAccount;

Cout << “Enter receiver account number: “;

Cin >> receiverAccount;

Auto senderIt = find\_if(accounts.begin(), accounts.end(),

[&senderAccount](const Account &account) {

Return account.account\_number == senderAccount;

});

If (senderIt != accounts.end()) {

Cout << “Enter payment amount: “;

Cin >> amount;

If (amount > senderIt->balance) {

Cout << “Insufficient funds.\n”;

} else {

Auto receiverIt = find\_if(accounts.begin(), accounts.end(),

[&receiverAccount](const Account &account) {

Return account.account\_number == receiverAccount;

});

If (receiverIt != accounts.end()) {

senderIt->balance -= amount;

receiverIt->balance += amount;

cout << “Payment successful.\n”;

} else {

Cout << “Receiver account not found.\n”;

}

}

} else {

Cout << “Sender account not found.\n”;

}

}

// Function to display account information

Void displayAccounts(const vector<Account> &accounts) {

Cout << “\nAccount Information:\n”;

For (const auto &account : accounts) {

Cout << “Account Number: “ << account.account\_number

<< “, Balance: “ << account.balance << “\n”;

}

Cout << “\n”;

}

Int main() {

Vector<Account> accounts;

Int choice;

Do {

Cout << “Banking System Menu:\n”;

Cout << “1. Create Account\n”;

Cout << “2. Withdrawal\n”;

Cout << “3. Deposit\n”;

Cout << “4. Payment\n”;

Cout << “5. Display Accounts\n”;

Cout << “0. Exit\n”;

Cout << “Enter your choice: “;

Cin >> choice;

Switch (choice) {

Case 1:

createAccount(accounts);

break;

case 2:

withdrawal(accounts);

break;

case 3:

deposit(accounts);

break;

case 4:

payment(accounts);

break;

case 5:

displayAccounts(accounts);

break;

case 0:

cout << “Exiting program. Goodbye!\n”;

break;

default:

cout << “Invalid choice. Please enter a valid option.\n”;

}

} while (choice != 0);

Return 0;

}