Banking System

```
#include<iostream>
#include<fstream>
#include<cstdlib>
#include<vector>
#include<map>
using namespace std;


#define MIN_BALANCE 500
class InsufficientFunds{};

class Account
{
private:
    long accountNumber;
    string firstName;
    string lastName;
    float balance;
    static long NextAccountNumber;
public:
    Account(){}
    Account(string fname,string lname,float balance);
    long getAccNo(){return accountNumber;}
    string getFirstName(){return firstName;}
    string getLastName(){return lastName;}
    float getBalance(){return balance;}

    void Deposit(float amount);
    void Withdraw(float amount);
    static void setLastAccountNumber(long accountNumber);
    static long getLastAccountNumber();
    friend ofstream & operator<<(ofstream &ofs,Account &acc);
    friend ifstream & operator>>(ifstream &ifs,Account &acc);
    friend ostream & operator<<(ostream &os,Account &acc);
};
long Account::NextAccountNumber=0;

class Bank
{
private:
    map<long,Account> accounts;
public:
    Bank();
    Account OpenAccount(string fname,string lname,float balance);
    Account BalanceEnquiry(long accountNumber);
    Account Deposit(long accountNumber,float amount);
    Account Withdraw(long accountNumber,float amount);
    void CloseAccount(long accountNumber);
    void ShowAllAccounts();
    ~Bank();
};

int main()
{
    Bank b;
    Account acc;

    int choice;
    string fname,lname;
```

```
long accountNumber;
float balance;
float amount;
cout<<"***Banking System***"<<endl;
do
{
    cout<<"\n\tSelect one option below ";
    cout<<"\n\t1 Open an Account";
    cout<<"\n\t2 Balance Enquiry";
    cout<<"\n\t3 Deposit";
    cout<<"\n\t4 Withdrawal";
    cout<<"\n\t5 Close an Account";
    cout<<"\n\t6 Show All Accounts";
    cout<<"\n\t7 Quit";
    cout<<"\nEnter your choice: ";
    cin>>choice;
    switch(choice)
    {
        case 1:
            cout<<"Enter First Name: ";
            cin>>fname;
            cout<<"Enter Last Name: ";
            cin>>lname;
            cout<<"Enter initil Balance: ";
            cin>>balance;
            acc=b.OpenAccount(fname,lname,balance);
            cout<<endl<<"Congradulation Account is Created"<<endl;
            cout<<acc;
            break;
        case 2:
            cout<<"Enter Account Number:";
            cin>>accountNumber;
            acc=b.BalanceEnquiry(accountNumber);
            cout<<endl<<"Your Account Details"<<endl;
            cout<<acc;
            break;
        case 3:
            cout<<"Enter Account Number:";
            cin>>accountNumber;
            cout<<"Enter Balance:";
            cin>>amount;
            acc=b.Deposit(accountNumber, amount);
            cout<<endl<<"Amount is Deposited"<<endl;
            cout<<acc;

            break;
        case 4:
            cout<<"Enter Account Number:";
            cin>>accountNumber;
            cout<<"Enter Balance:";
            cin>>amount;
            acc=b.Withdraw(accountNumber, amount);
            cout<<endl<<"Amount Withdrawn"<<endl;
            cout<<acc;
            break;
        case 5:
            cout<<"Enter Account Number:";
            cin>>accountNumber;
            b.CloseAccount(accountNumber);
            cout<<endl<<"Account is Closed"<<endl;
            cout<<acc;
        case 6:
            b.ShowAllAccounts();
            break;
        case 7: break;
        default:
```

```
                cout<<"\nEnter corret choice";
                exit(0);
        }
    }while(choice!=7);

    return 0;
}

Account::Account(string fname,string lname,float balance)
{
    NextAccountNumber++;
    accountNumber=NextAccountNumber;
    firstName=fname;
    lastName=lname;
    this->balance=balance;
}
void Account::Deposit(float amount)
{
    balance+=amount;
}
void Account::Withdraw(float amount)
{
    if(balance-amount<MIN_BALANCE)
        throw InsufficientFunds();
    balance-=amount;
}
void Account::setLastAccountNumber(long accountNumber)
{
    NextAccountNumber=accountNumber;
}
long Account::getLastAccountNumber()
{
    return NextAccountNumber;
}
ofstream & operator<<(ofstream &ofs,Account &acc)
{
    ofs<<acc.accountNumber<<endl;
    ofs<<acc.firstName<<endl;
    ofs<<acc.lastName<<endl;
    ofs<<acc.balance<<endl;
    return ofs;
}
ifstream & operator>>(ifstream &ifs,Account &acc)
{
    ifs>>acc.accountNumber;
    ifs>>acc.firstName;
    ifs>>acc.lastName;
    ifs>>acc.balance;
    return ifs;

}
ostream & operator<<(ostream &os,Account &acc)
{
    os<<"First Name:"<<acc.getFirstName()<<endl;
    os<<"Last Name:"<<acc.getLastName()<<endl;
    os<<"Account Number:"<<acc.getAccNo()<<endl;
    os<<"Balance:"<<acc.getBalance()<<endl;
    return os;
}


Bank::Bank()
{

    Account account;
    ifstream infile;
```

```
    infile.open("Bank.data");
    if(!infile)
    {
        //cout<<"Error in Opening! File Not Found!!"<<endl;
        return;
    }
    while(!infile.eof())
    {
        infile>>account;
        accounts.insert(pair<long,Account>(account.getAccNo(),account));
    }
    Account::setLastAccountNumber(account.getAccNo());

    infile.close();

}

Account Bank::OpenAccount(string fname,string lname,float balance)
{
    ofstream outfile;
    Account account(fname,lname,balance);
    accounts.insert(pair<long,Account>(account.getAccNo(),account));

    outfile.open("Bank.data", ios::trunc);

    map<long,Account>::iterator itr;
    for(itr=accounts.begin();itr!=accounts.end();itr++)
    {
        outfile<<itr->second;
    }
    outfile.close();
    return account;
}
Account Bank::BalanceEnquiry(long accountNumber)
{
    map<long,Account>::iterator itr=accounts.find(accountNumber);
    return itr->second;
}
Account Bank::Deposit(long accountNumber,float amount)
{
    map<long,Account>::iterator itr=accounts.find(accountNumber);
    itr->second.Deposit(amount);
    return itr->second;
}
Account Bank::Withdraw(long accountNumber,float amount)
{
    map<long,Account>::iterator itr=accounts.find(accountNumber);
    itr->second.Withdraw(amount);
    return itr->second;
}
void Bank::CloseAccount(long accountNumber)
{
    map<long,Account>::iterator itr=accounts.find(accountNumber);
    cout<<"Account Deleted"<<itr->second;
    accounts.erase(accountNumber);
}
void Bank::ShowAllAccounts()
{
    map<long,Account>::iterator itr;
    for(itr=accounts.begin();itr!=accounts.end();itr++)
    {
        cout<<"Account "<<itr->first<<endl<<itr->second<<endl;
    }
}
Bank::~Bank()
{
```

```
    ofstream outfile;
    outfile.open("Bank.data", ios::trunc);

    map<long,Account>::iterator itr;
    for(itr=accounts.begin();itr!=accounts.end();itr++)
    {
        outfile<<itr->second;
    }
    outfile.close();
}
```