```cpp
                                                                    //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;
```

```
 79   cout<<"Enter initil Balance: ";
 80   cin>>balance;
 81    acc=b.OpenAccount(fname,lname,balance);
 82    cout<<endl<<"Congradulation Account is Created"<<endl;
 83    cout<<acc;
 84   break;
 85    case 2:
 86    cout<<"Enter Account Number:";
 87   cin>>accountNumber;
 88    acc=b.BalanceEnquiry(accountNumber);
 89    cout<<endl<<"Your Account Details"<<endl;
 90    cout<<acc;
 91   break;
 92    case 3:
 93    cout<<"Enter Account Number:";
 94   cin>>accountNumber;
 95   cout<<"Enter Balance:";
 96   cin>>amount;
 97    acc=b.Deposit(accountNumber, amount);
 98    cout<<endl<<"Amount is Deposited"<<endl;
 99    cout<<acc;
100   break;
101    case 4:
102    cout<<"Enter Account Number:";
103   cin>>accountNumber;
104   cout<<"Enter Balance:";
105   cin>>amount;
106    acc=b.Withdraw(accountNumber, amount);
107    cout<<endl<<"Amount Withdrawn"<<endl;
108    cout<<acc;
109   break;
110    case 5:
111    cout<<"Enter Account Number:";
112   cin>>accountNumber;
113    b.CloseAccount(accountNumber);
114    cout<<endl<<"Account is Closed"<<endl;
115    cout<<acc;
116    case 6:
117    b.ShowAllAccounts();
118    break;
119    case 7: break;
120    default:
121    cout<<"\nEnter correct choice";
122   exit(0);
123    }
124    }while(choice!=7);
125
126    return 0;
127   }
128   Account::Account(string fname,string lname,float balance)
129   {
130    NextAccountNumber++;
131    accountNumber=NextAccountNumber;
132    firstName=fname;
133    lastName=lname;
134    this->balance=balance;
135   }
136   void Account::Deposit(float amount)
137   {
138    balance+=amount;
139   }
140   void Account::Withdraw(float amount)
141   {
142    if(balance-amount<MIN_BALANCE)
143    throw InsufficientFunds();
144    balance-=amount;
145   }
146   void Account::setLastAccountNumber(long accountNumber)
147   {
148    NextAccountNumber=accountNumber;
149   }
150   long Account::getLastAccountNumber()
151   {
152    return NextAccountNumber;
153   }
154   ofstream & operator<<(ofstream &ofs,Account &acc)
155   {
156    ofs<<acc.accountNumber<<endl;
```

```
157     ofs<<acc.firstName<<endl;
158     ofs<<acc.lastName<<endl;
159     ofs<<acc.balance<<endl;
160     return ofs;
161    }
162    ifstream & operator>>(ifstream &ifs,Account &acc)
163    {
164     ifs>>acc.accountNumber;
165     ifs>>acc.firstName;
166     ifs>>acc.lastName;
167     ifs>>acc.balance;
168     return ifs;
169
170    }
171    ostream & operator<<(ostream &os,Account &acc)
172    {
173     os<<"First Name:"<<acc.getFirstName()<<endl;
174     os<<"Last Name:"<<acc.getLastName()<<endl;
175     os<<"Account Number:"<<acc.getAccNo()<<endl;
176     os<<"Balance:"<<acc.getBalance()<<endl;
177     return os;
178    }
179    Bank::Bank()
180    {
181
182     Account account;
183     ifstream infile;
184     infile.open("Bank.data");
185     if(!infile)
186     {
187     //cout<<"Error in Opening! File Not Found!!"<<endl;
188     return;
189     }
190     while(!infile.eof())
191     {
192     infile>>account;
193     accounts.insert(pair<long,Account>(account.getAccNo(),account));
194     }
195     Account::setLastAccountNumber(account.getAccNo());
196
197     infile.close();
198
199    }
200    Account Bank::OpenAccount(string fname,string lname,float balance)
201    {
202     ofstream outfile;
203     Account account(fname,lname,balance);
204     accounts.insert(pair<long,Account>(account.getAccNo(),account));
205
206     outfile.open("Bank.data", ios::trunc);
207
208     map<long,Account>::iterator itr;
209     for(itr=accounts.begin();itr!=accounts.end();itr++)
210     {
211     outfile<<itr->second;
212     }
213     outfile.close();
214     return account;
215    }
216    Account Bank::BalanceEnquiry(long accountNumber)
217    {
218     map<long,Account>::iterator itr=accounts.find(accountNumber);
219     return itr->second;
220    }
221    Account Bank::Deposit(long accountNumber,float amount)
222    {
223     map<long,Account>::iterator itr=accounts.find(accountNumber);
224     itr->second.Deposit(amount);
225     return itr->second;
226    }
227    Account Bank::Withdraw(long accountNumber,float amount)
228    {
229     map<long,Account>::iterator itr=accounts.find(accountNumber);
230     itr->second.Withdraw(amount);
231     return itr->second;
232    }
233    void Bank::CloseAccount(long accountNumber)
234    {
```

```
 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();
}
```