#include <iostream>

#include <vector>

#include <ctime>

#include <fstream>

#include <iomanip>

using namespace std;

class Person{

private:

string firstName, lastName;

public:

Person(){

firstName = "";

lastName = "";

}

Person(string fn, string ln){

firstName = fn;

lastName = ln;

}

void setFirstName(string fn){

firstName = fn;

}

string getFirstName(){

return firstName;

}

void setLastName(string ln){

lastName = ln;

}

string getLastName(){

return lastName;

}

// will return firstname + last name

string getFullName(){

return firstName + " " + lastName;

}

};

class Date{

private:

int day, month, year;

public:

Date(){}

Date(int d, int m, int y){

day = d;

month = m;

year = y;

}

void setDay(int d){

day = d;

}

int getDay(){

return day;

}

void setMonth(int m){

month = m;

}

int getMonth(){

return month;

}

void setYear(int y){

year = y;

}

int getYear(){

return year;

}

void setCurrentDate(){

time\_t c = time(0);

tm \*current = localtime(&c);

setDay(current->tm\_mday);

setMonth(current->tm\_mon + 1);

setYear(current->tm\_year + 1900);

}

void print(){

cout << day << ":" << month << ":" << year << endl;

}

int getDifference(Date d1, Date d2){

int n1 = d1.year\*365 + d2.day;

for (int i = 0; i < d1.month-1; i++)

n1 += 30;

int n2 = d1.year\*365 + d1.day;

for (int i = 0; i < d2.month-1; i++)

n2 += 30;

return (n2 - n1);

}

};

class Book {

private:

int libraryCode, ISBN, rack, shelf;

bool isLoaned;

string title, description, publisher;

vector<string> authors;

Date publishDate;

public:

Book(){}

Book(int lc, int ibs, bool il, string t, string d, string p, vector<string> a, Date pd){

libraryCode = lc;

ISBN = ibs;

isLoaned = il;

title = t;

description = d;

publisher = p;

authors = a;

publishDate = pd;

setRack(lc / 100);

setShelf(lc % 100);

}

void setLibraryCode(int lc){

libraryCode = lc;

}

int getLibraryCode(){

return libraryCode;

}

void setISBN(int ibs){

ISBN = ibs;

}

int getISBN(){

return ISBN;

}

void setRack(int r){

rack = r;

}

int getRack(){

return rack;

}

void setShelf(int s){

shelf = s;

}

int getShelf(){

return shelf;

}

void setIsLoaned(bool il){

isLoaned = il;

}

bool getIsLoaned(){

return isLoaned;

}

void setTitle(string t){

title = t;

}

string getTitle(){

return title;

}

void setDesctiption(string d){

description = d;

}

string getDescription(){

return description;

}

void setPublisher(string p){

publisher = p;

}

string getPublisher(){

return publisher;

}

void setAuthors(vector<string> a){

authors = a;

}

vector<string> getAuthors(){

return authors;

}

void setPublishDate(Date pd){

publishDate = pd;

}

Date getPublishDate(){

return publishDate;

}

void print(){

cout << "\n------------ \"" << title << "\" book information ------------" << endl;

cout << "Description: " << description << endl;

cout << "Authors: ";

for(int i=0; i<authors.size(); i++)

cout << authors[i] << ", ";

cout << endl;

cout << "Publisher: " << publisher << endl;

cout << "Publish date: " ;

publishDate.print();

cout << endl;

cout << "Library code, ISBN, rack, shelf: " << libraryCode << ", " << ISBN << ", " << rack << ", " << shelf << endl;

cout << "Is available? ";

if(!isLoaned)

cout << "Yes" << endl;

else

cout << "No" << endl;

cout << endl;

}

};

class Customer : public Person{

private:

int accountNumber;

double fine;

vector<Book> issuedBooks;

vector<Date> issuedDates;

vector<int> loanedFor;

bool isRestricted;

public:

Customer() : Person(){

accountNumber = 0;

fine = 0;

setIsRestricted();

}

Customer(string fn, string ln, int an, double f, vector<Book> ib, vector<Date> id, vector<int> lf) : Person(fn, ln){

accountNumber = an;

fine = f;

issuedBooks = ib;

issuedDates = id;

loanedFor = lf;

setIsRestricted();

}

void setIsRestricted(){

if(fine > 0)

isRestricted = true;

else

isRestricted = false;

}

bool getIsRestricted(){

return isRestricted;

}

void setAccountNumber(int an){

accountNumber = an;

}

int getAccountNumber(){

return accountNumber;

}

void setIssuedBooks(vector<Book> ib){

issuedBooks = ib;

}

vector<Book> getIssuedBooks(){

return issuedBooks;

}

void setIssuedDates(vector<Date> id){

issuedDates = id;

}

vector<Date> getIssuedDates(){

return issuedDates;

}

void setLoanedFor(vector<int> lf){

loanedFor = lf;

}

vector<int> getLoanedFor(){

return loanedFor;

}

void setFine(double f){

fine = f;

}

void incrementFine(){

fine \*= 0.1;

}

double getFine(){

return fine;

}

int getTotalIssuedBooks(){

return issuedBooks.size();

}

// append a book to issued books

void addBook(Book b){

issuedBooks.push\_back(b);

}

// append current date in issued dates

void addCurrentDate(){

Date d;

d.setCurrentDate();

issuedDates.push\_back(d);

}

// append no. of days book requested to load for

void addLoanedFor(int lf){

loanedFor.push\_back(lf);

}

// return book index using its library code

int getBookIndex(int libraryCode){

int i;

for(i=0; i<issuedBooks.size(); i++){

if(issuedBooks[i].getLibraryCode() == libraryCode)

break;

}

return i;

}

// remove loaned for days using index

void removeLoanedFor(int i){

loanedFor.erase(loanedFor.begin() + i);

}

// remove issued date using index

void removeIssuedDate(int i){

issuedDates.erase(issuedDates.begin() + i);

}

// remove book using index

void removeBook(int i){

issuedBooks.erase(issuedBooks.begin() + i);

}

// remove fine to zero

void removeFine(){

fine = 0;

setIsRestricted();

}

// will check if customer own library code

bool hasBook(int libraryCode){

for(int i=0; i<issuedBooks.size(); i++){

if(issuedBooks[i].getLibraryCode() == libraryCode)

return true;

}

return false;

}

void print(){

cout << "\n------------ " << getFullName() << " information ------------" << endl;

cout << "Account number: " << accountNumber << endl;

cout << "Fine: " << fine << endl;

for(int i=0; i<issuedBooks.size(); i++)

issuedBooks[i].print();

}

};

class Staff : public Person{

public:

Staff(): Person(){}

Staff(string fn, string ln): Person(fn, ln){};

};

class Student : public Person{

public:

Student(): Person(){}

Student(string fn, string ln): Person(fn, ln){};

};

class Faculty : public Person{

public:

Faculty(): Person(){}

Faculty(string fn, string ln): Person(fn, ln){};

};

class LibraryStaff : public Person{

public:

LibraryStaff(): Person(){}

LibraryStaff(string fn, string ln): Person(fn, ln){};

// return customer searched by account number or first name or last name

vector<Customer> searchCustomer(int accountNumber, string fn, string ln, vector<Customer> customers){

vector<Customer> temp;

for(int i=0; i<customers.size(); i++){

if(customers[i].getAccountNumber() == accountNumber || customers[i].getFirstName() == fn || customers[i].getLastName() == ln)

temp.push\_back(customers[i]);

}

return temp;

}

};

// read customers file and store them in customers vector

void readCustomers(vector<Customer> &customers){

ifstream customerFile("customers.dat");

if(!customerFile){

cout << "ERROR!!! Can not open customers.txt file !!! " << endl;

exit(0);

}

int accountNumber;

double fine;

string firstName, lastName;

while(!customerFile.eof()){

customerFile >> accountNumber;

customerFile >> firstName;

customerFile >> lastName;

customerFile >> fine;

Customer c;

c.setAccountNumber(accountNumber);

c.setFirstName(firstName);

c.setLastName(lastName);

c.setFine(fine);

c.setIsRestricted();

customers.push\_back(c);

}

if(customers[customers.size()-1].getAccountNumber() == customers[customers.size()-2].getAccountNumber())

customers.pop\_back();

customerFile.close();

cout << "--- Customers.txt file read successfully ---" << endl;

}

// write customers file from customers vector

void writeCustomers(vector<Customer> customers){

ofstream customerFile("customers.dat");

if(!customerFile){

cout << "ERROR!!! Can not open customers.txt file !!! " << endl;

return;

}

for(int i=0; i<customers.size(); i++){

customerFile << setw(5) << customers[i].getAccountNumber();

customerFile << setw(10) << customers[i].getFirstName();

customerFile << setw(10) << customers[i].getLastName();

customerFile << setw(10) << customers[i].getFine();

customerFile << endl;

}

customerFile.close();

cout << "--- Customers.txt file write successfully ---" << endl;

}

// read books file and store them in books vector

void readBooks(vector<Book> &books){

ifstream booksFile("books.dat");

if(!booksFile){

cout << "ERROR!!! Can not open books.txt file !!! " << endl;

exit(0);

}

int libraryCode, ISBN, totalAuthors, day, month, year;

bool isLoaned;

char title[20], description[30], publisher[20];

vector<string> authors;

Date publishDate;

while(!booksFile.eof()){

authors.clear();

booksFile >> libraryCode;

booksFile >> ISBN;

booksFile >> isLoaned;

booksFile.ignore();

booksFile.read(title, 20);

title[19] = '\0';

booksFile.ignore();

booksFile.read(description, 30);

description[29] = '\0';

booksFile.ignore();

booksFile.read(publisher, 20);

publisher[19] = '\0';

booksFile >> totalAuthors;

for(int i=0; i<totalAuthors; i++){

char authorName[15];

booksFile.ignore();

booksFile.read(authorName, 15);

authorName[14] = '\0';

authors.push\_back(authorName);

}

booksFile >> day;

booksFile >> month;

booksFile >> year;

Book b(libraryCode, ISBN, isLoaned, title, description, publisher, authors, Date(day, month, year));

books.push\_back(b);

}

books.pop\_back();

booksFile.close();

cout << "--- books.txt file read successfully ---" << endl;

}

// write books file from books vector

void writeBooks(vector<Book> books){

ofstream booksFile("books.dat");

if(!booksFile){

cout << "ERROR!!! Can not open books.txt file !!! " << endl;

return;

}

for(int i=0; i<books.size(); i++){

booksFile << setw(4) << books[i].getLibraryCode();

booksFile << " ";

booksFile << setw(8) << books[i].getISBN();

booksFile << " ";

booksFile << setw(1) << books[i].getIsLoaned();

booksFile << setw(20) << books[i].getTitle();

booksFile << " ";

booksFile << setw(30) << books[i].getDescription();

booksFile << " ";

booksFile << setw(20) << books[i].getPublisher();

booksFile << " ";

booksFile << setw(1) << books[i].getAuthors().size();

for(int j=0; j<books[i].getAuthors().size(); j++){

booksFile << setw(15) << books[i].getAuthors()[j];

booksFile << " ";

}

booksFile << books[i].getPublishDate().getDay();

booksFile << " ";

booksFile << books[i].getPublishDate().getMonth();

booksFile << " ";

booksFile << books[i].getPublishDate().getYear();

booksFile << endl;

}

booksFile.close();

cout << "--- books.txt file write successfully ---" << endl;

}

// read issued detail file and store them in issued detail vector

void readIssuedBooksRecords(vector<Customer> &customers, vector<Book> books){

ifstream customerFile("issued\_books\_records.dat");

int accountNumber, totalIssued, libraryCode, day, month, year, loanedForDays;

vector<Book> issuedBooks;

vector<Date> issuedDates;

vector<int> loanedFor;

if(!customerFile){

cout << "ERROR!!! Can not open issued\_books\_records.txt file !!! " << endl;

exit(0);

}

while(!customerFile.eof()){

issuedBooks.clear();

issuedDates.clear();

loanedFor.clear();

customerFile >> accountNumber;

customerFile >> totalIssued;

for(int i=0; i<totalIssued; i++){

customerFile >> libraryCode;

for(int j=0; j<books.size(); j++){

if(books[j].getLibraryCode() == libraryCode)

issuedBooks.push\_back(books[j]);

}

customerFile >> day;

customerFile >> month;

customerFile >> year;

issuedDates.push\_back(Date(day, month, year));

customerFile >> loanedForDays;

loanedFor.push\_back(loanedForDays);

}

for(int i=0; i<customers.size(); i++){

if(customers[i].getAccountNumber() == accountNumber){

customers[i].setIssuedDates(issuedDates);

customers[i].setIssuedBooks(issuedBooks);

customers[i].setLoanedFor(loanedFor);

}

}

}

//customers.pop\_back();

customerFile.close();

cout << "--- issued\_books\_records.txt file read successfully ---" << endl;

}

// write issued detail file from issued detail vector

void writeIssuedBooksRecords(vector<Customer> customers, vector<Book> books){

ofstream customerFile("issued\_books\_records.dat");

if(!customerFile){

cout << "ERROR!!! Can not open issued\_books\_records.txt file !!! " << endl;

return;

}

for(int i=0; i<customers.size(); i++){

customerFile << customers[i].getAccountNumber();

customerFile << " ";

customerFile << customers[i].getIssuedBooks().size();

customerFile << " ";

for(int j=0; j<customers[i].getIssuedBooks().size(); j++){

customerFile << customers[i].getIssuedBooks()[j].getLibraryCode();

customerFile << " ";

customerFile << customers[i].getIssuedDates()[j].getDay();

customerFile << " ";

customerFile << customers[i].getIssuedDates()[j].getMonth();

customerFile << " ";

customerFile << customers[i].getIssuedDates()[j].getYear();

customerFile << " ";

customerFile << customers[i].getLoanedFor()[j];

customerFile << " ";

}

customerFile << endl;

}

customerFile.close();

cout << "--- issued\_books\_records.txt file write successfully ---" << endl;

}

// loan a book

void loanBook(int accountNumber, int libraryCode, int loanedFor, vector<Customer> &customers, vector<Book> &books){

int customerIndex = 0;

for(customerIndex = 0; customerIndex<customers.size(); customerIndex++){

if(customers[customerIndex].getAccountNumber() == accountNumber)

break;

}

int bookIndex=0;

for(bookIndex=0; bookIndex<books.size(); bookIndex++){

if(books[bookIndex].getLibraryCode() == libraryCode)

break;

}

if(customers[customerIndex].getIsRestricted()){

cout << "!!! "<< customers[customerIndex].getFullName() << " has pending fine, can't borrow new book !!!" << endl << endl;

return;

}

else if(customers[customerIndex].getIssuedBooks().size() == 2){

cout << "!!! " << customers[customerIndex].getFullName() << " already have maximum allowed books loaned, can't borrow new book !!!" << endl << endl;

return;

}

else if(books[bookIndex].getIsLoaned()){

cout << "!!! " << books[bookIndex].getTitle() << " is already been loaned by someone, wait till return of book !!!" << endl << endl;

return;

}

customers[customerIndex].addBook(books[bookIndex]);

customers[customerIndex].addCurrentDate();

customers[customerIndex].addLoanedFor(loanedFor);

books[bookIndex].setIsLoaned(true);

cout << "\n--- " << books[bookIndex].getTitle() << " is assigned to " << customers[customerIndex].getFullName() << " for " << loanedFor << " days ---" << endl << endl;

}

// pay fine

void payFine(int accountNumber, vector<Customer> &customers){

int customerIndex = 0;

for(customerIndex = 0; customerIndex<customers.size(); customerIndex++){

if(customers[customerIndex].getAccountNumber() == accountNumber)

break;

}

customers[customerIndex].removeFine();

cout << "--- " << customers[customerIndex].getFullName() << " fine paid successfully ---" << endl << endl;

}

// return book

void returnBook(int accountNumber, int libraryCode, vector<Customer> &customers, vector<Book> &books){

int customerIndex = 0;

for(customerIndex = 0; customerIndex<customers.size(); customerIndex++){

if(customers[customerIndex].getAccountNumber() == accountNumber)

break;

}

int bookIndex=0;

for(bookIndex=0; bookIndex<books.size(); bookIndex++){

if(books[bookIndex].getLibraryCode() == libraryCode)

break;

}

if(!customers[customerIndex].hasBook(libraryCode)){

cout << "!!! "<< customers[customerIndex].getFullName() << " doest not loaned " << books[bookIndex].getTitle() << " !!!" << endl << endl;

return;

}

Date temp;

temp.setCurrentDate();

int index = customers[customerIndex].getBookIndex(libraryCode);

int difference = temp.getDifference(customers[customerIndex].getIssuedDates()[index], temp);

int loanedFor = customers[customerIndex].getLoanedFor()[index];

books[bookIndex].setIsLoaned(false);

customers[customerIndex].removeBook(index);

customers[customerIndex].removeLoanedFor(index);

customers[customerIndex].removeIssuedDate(index);

if(difference > loanedFor){

difference -= loanedFor;

int fine = 10 + customers[customerIndex].getFine();

for(int i=0; i<difference; i++)

fine += (fine \* 0.1);

customers[customerIndex].setFine(fine);

customers[customerIndex].setIsRestricted();

cout << "!!! Late return of " << difference << " days, pending fine = " << customers[customerIndex].getFine() << " !!!" << endl<<endl;

return;

}

cout << "--- " << books[bookIndex].getTitle() << " is return from " << customers[customerIndex].getFullName() << " successfully ---" << endl << endl;

}

// search book

vector<Book> searchBook(int libraryCode, int ISBN, string title, string author, bool searchViaAvailability, bool isAvailable, vector<Book> books){

vector<Book> temp;

for(int i=0; i<books.size(); i++){

for(int j=0; j<books[i].getAuthors().size(); j++){

//cout << "\"" << books[i].getAuthors()[j] << "\"" << endl;

if(books[i].getAuthors()[j] == author){

temp.push\_back(books[i]);

}

}

}

for(int i=0; i<books.size(); i++){

if(books[i].getLibraryCode() == libraryCode || books[i].getISBN() == ISBN || books[i].getTitle() == title)

temp.push\_back(books[i]);

}

if(searchViaAvailability){

for(int i=0; i<books.size(); i++){

if(books[i].getIsLoaned() == !isAvailable)

temp.push\_back(books[i]);

}

}

return temp;

}

// print a generic vector

template <class T>

void printVector(vector<T> v){

for(int i=0; i<v.size(); i++)

v[i].print();

}

int main()

{

vector<Customer> customers;

vector<Book> books;

LibraryStaff LF;

readCustomers(customers);

readBooks(books);

readIssuedBooksRecords(customers, books);

cout << endl;

cout << "----------- Requesting to loan a book while fine is pending ------------" << endl;

loanBook(33, 2253, 14, customers, books);

payFine(33, customers);

loanBook(44, 2233, 30, customers, books);

payFine(44, customers);

loanBook(33, 2253, 14, customers, books);

cout << "------------------------------------------------------------------------" << endl;

cout << "----------- Requesting to loan a book while already have maximum books loan ------------" << endl;

loanBook(44, 2254, 20, customers, books);

cout << "----------------------------------------------------------------------------------------" << endl;

cout << "----------- Returning a book ------------" << endl;

returnBook(44, 2233, customers, books);

cout << "-----------------------------------------" << endl;

cout << "----------- Requesting to loan a book after returning previous ------------" << endl;

loanBook(44, 2254, 20, customers, books);

cout << "---------------------------------------------------------------------------" << endl;

cout << "----------- Returning a book ------------" << endl;

returnBook(33, 2253, customers, books);

cout << "-----------------------------------------" << endl;

cout << "----------- Returning a book ------------" << endl;

returnBook(55, 7889, customers, books);

cout << "-----------------------------------------" << endl;

cout << "----------- Searching book via library code ------------" << endl;

printVector(searchBook(7889, 0, "", "", false, false, books));

cout << "--------------------------------------------------------" << endl;

cout << "----------- Searching book via ISBN number ------------" << endl;

printVector(searchBook(0, 22501110, "", "", false, false, books));

cout << "-------------------------------------------------------" << endl;

cout << "----------- Searching book via book title ------------" << endl;

printVector(searchBook(0, 0, "book two ", "", false, false, books));

cout << "------------------------------------------------------" << endl;

cout << "----------- Searching book via author name ------------" << endl;

printVector(searchBook(0, 0, "", "author 2278 1 ", false, false, books));

cout << "--------------------------------------------------------" << endl;

cout << "----------- Searching book via availability ------------" << endl;

printVector(searchBook(0, 0, "", "", true, true, books));

cout << "--------------------------------------------------------" << endl;

cout << "----------- Searching customer via account number ------------" << endl;

printVector(LF.searchCustomer(33, "", "", customers));

cout << "--------------------------------------------------------" << endl;

cout << "----------- Searching customer via first name ------------" << endl;

printVector(LF.searchCustomer(0, "win", "", customers));

cout << "--------------------------------------------------------" << endl;

cout << "----------- Searching customer via last name ------------" << endl;

printVector(LF.searchCustomer(0, "", "carrey", customers));

cout << "--------------------------------------------------------" << endl;

cout << endl;

writeCustomers(customers);

writeBooks(books);

writeIssuedBooksRecords(customers, books);

return 0;

}

