**Assignment 5**

Q1.

CODE:

//operator overloading in cpp(4)

#include<iostream>

using namespace std;

#define nstu student\_list::no\_of\_students

#define nsub subject\_list::no\_of\_subjects

#define nsubstu subject::no\_of\_students

class subject{

static int no\_of\_students;

int code;

char name[11];

public:

void displayStudents(int);

friend class subject\_list;

friend class student;

friend class student\_list;

};

class subject\_list{

static int no\_of\_subjects;

subject list[100];

public:

void addSubject();

friend class student;

friend class student\_list;

};

class student{

int roll;

char name[31];

char phone[11];

subject\_list subs;

public:

friend class student\_list;

};

class student\_list{

static int no\_of\_students;

student list[100];

public:

void addStudent();

void displaySub(int);

void displayStudents(string);

};

int student\_list::no\_of\_students = 0;

int subject\_list::no\_of\_subjects=0;

int subject::no\_of\_students=0;

void student\_list::addStudent(){

if(nstu>100){

cout<<"STUDENT LIMIT REACHED. PLEASE CONTACT DEAN.\n";

return;

}

++nstu;

int ch;

list[nstu].roll=nstu;

cout<<"Enter Student Details:\n";

cout<<"Name: ";

cin>>list[nstu].name;

cout<<"Phone No.: ";

cin>>list[nstu].phone;

do{

list[nstu].subs.addSubject();

cout<<"Press 1 to add another subject, 0 to end: ";

cin>>ch;

}while(ch!=0);

cout<<"Alloted Roll of Student: A00"<<list[nstu].roll<<"\n";

}

void subject\_list::addSubject(){

if(nsub>100){

cout<<"SUBJECT LIMIT EXCEEDED. PLEASE CONTACT DEAN.\n";

return;

}

++nsub;

list[nsub].code=nsub;

cout<<"Enter Subject Name: ";

cin>>list[nsub].name;

cout<<"Subject Code: S00"<<list[nsub].code<<"\n";

}

void student\_list::displaySub(int a){

if(a<=0 || a>nstu){

cout<<"Invalid Roll Number!\n";

return;

}

cout<<"Subjects chosen by the student are:\n";

for(int i=1;i<=list[a].subs.no\_of\_subjects;i++)

cout<<list[a].subs.list[i].name<<"\n";

}

void student\_list::displayStudents(string a){

// if(a<=0 || a>nstu){

// cout<<"Invalid Subject Code!\n";

// return;

// }

cout<<"Students are:\n";

for(int i=1;i<=nstu;i++){

for(int j=1;j<=nsub;j++)

if(list[i].subs.list[j].name==a)

cout<<list[i].name<<"\n";

}

}

int main(){

student\_list ob;

int ch;

string stu\_code;

string sub;

do{

cout<<"\n\*\*\*STUDENT ADMISSION SYSTEM\*\*\*\n\n";

cout<<"1. ADD STUDENT.\n";

cout<<"2. DISPLAY STUDENT'S SUBJECTS.\n";

cout<<"3. DISPLAY STUDENTS AGAINST A SUBJECT.\n";

cout<<"4. EXIT.\n";

cout<<"Enter Your Choice: ";

cin>>ch;

switch(ch){

case 1:

ob.addStudent();

break;

case 2:

cout<<"Enter Roll (in format A00X): ";

cin>>stu\_code;

ob.displaySub(stu\_code[3] - '0');

break;

case 3:

cout<<"Enter Subject Name: ";

cin>>sub;

ob.displayStudents(sub);

break;

case 4:

exit(0);

break;

default:

cout<<"INVALID CHOICE!!\n";

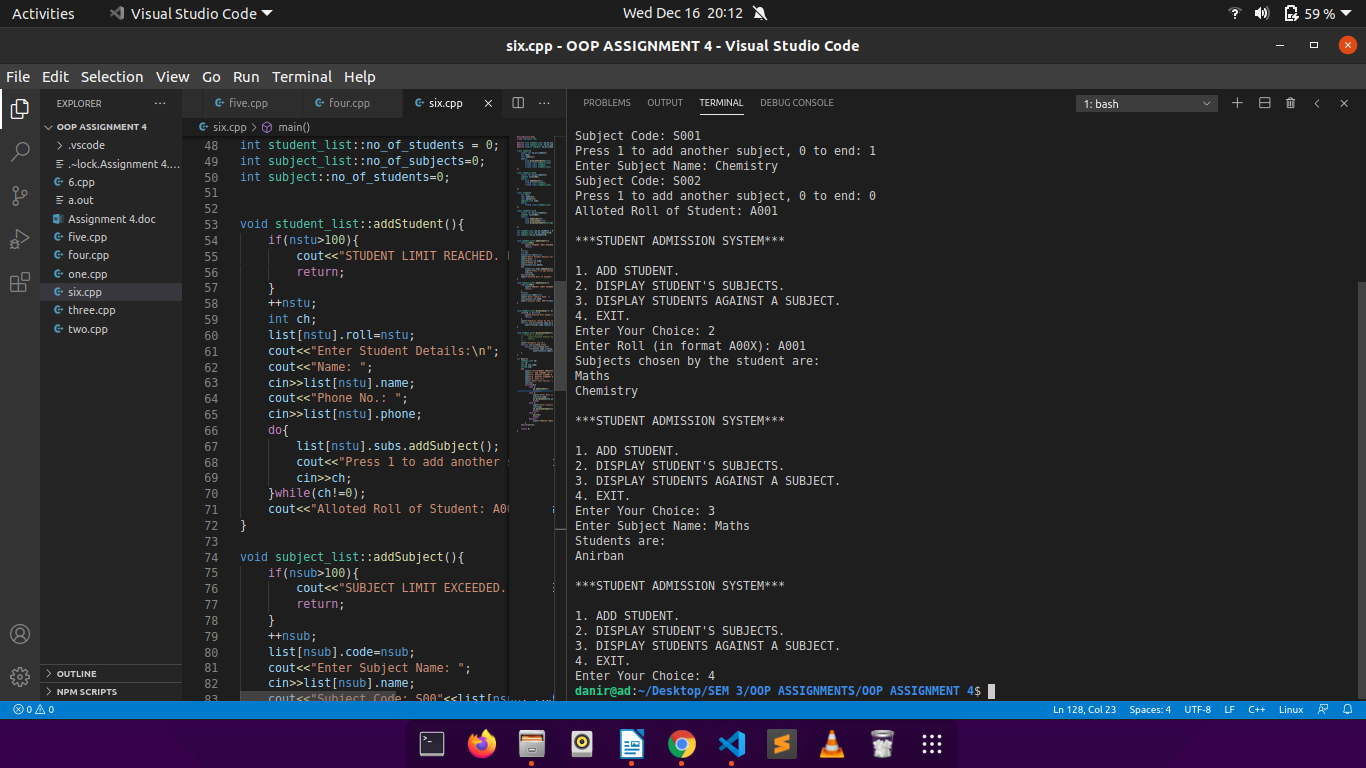
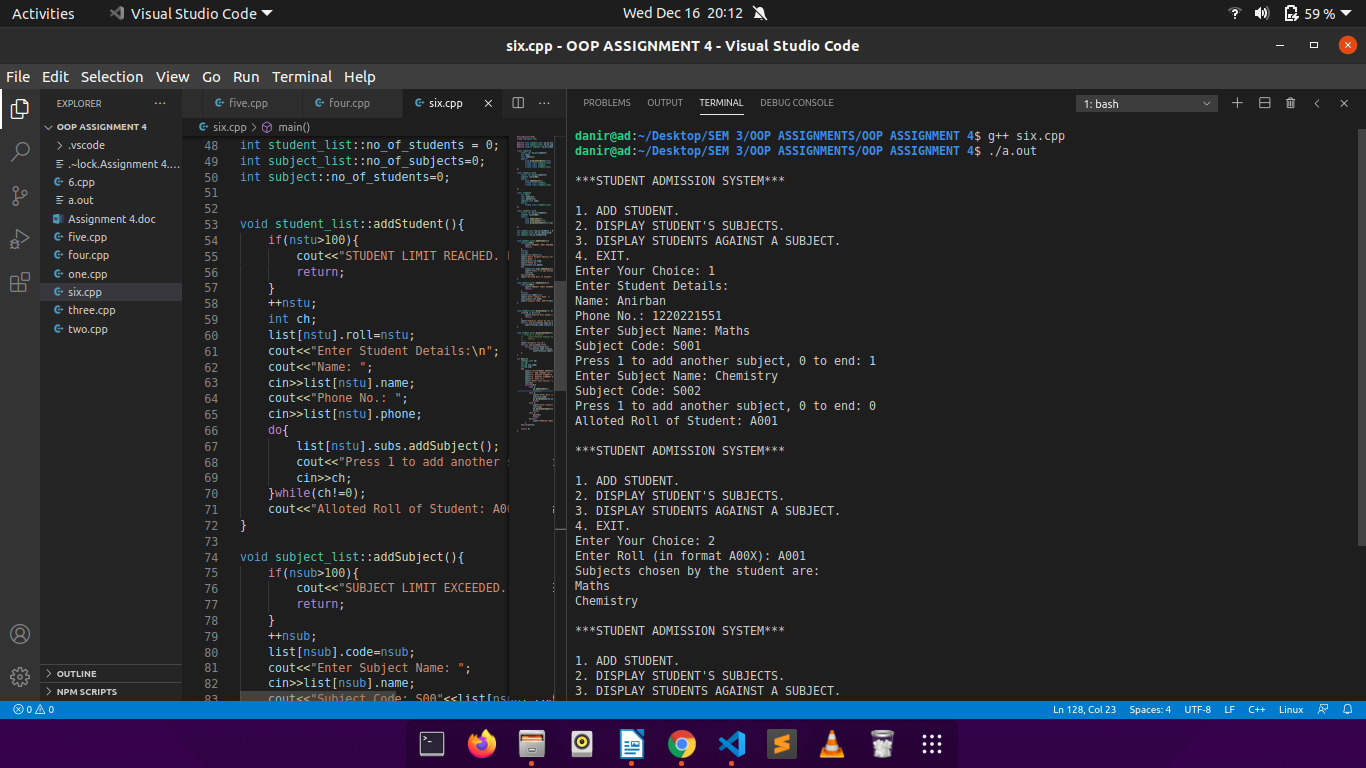
}

}while(ch!=4);

return 0;

}

OUTPUT:

****

Q2.

CODE:

#include<iostream>

using namespace std;

#define nob booklist::noofbooks

#define nos studentlist::noofstudents

#define nof facultylist::nooffaculties

#define notra transactionlist::nooftransactions

class booklist;

//book details

class book{

int serial;

int bookid;

char title[21];

char author[31];

char publisher[31];

int price;

public:

book(){

serial=1;

}

void getInfo(int id){

cout<<"Enter Book Details:\n";

cout<<"Title: ";

cin>>title;

cout<<"Author: ";

cin>>author;

cout<<"Publisher: ";

cin>>publisher;

cout<<"Price: ";

cin>>price;

id++;

bookid=id;

cout<<"Book-ID: "<<bookid<<endl;

}

int returnid(){ //returns book id

return bookid;

}

int returnserial(){ //returns current number of copies

return serial;

}

friend class booklist;

friend class transactionlist;

};

//member details

class member{

protected:

int memberid;

char name[31];

char email[21];

char address[51];

int numberofbooks;

int borrowed[11];

public:

void getInfo(){

cout<<"Enter Member Details:\n";

cout<<"Name: ";

cin>>name;

cout<<"Email: ";

cin>>email;

cout<<"Address: ";

cin>>address;

}

int returnmemid(){ //returns member id

return memberid;

}

friend class transactionlist;

};

class student:public member{ //inherited from member

public:

student(){ //constructor

numberofbooks=0;

}

int checkIssue(){ //checks if anymore issue is possible

if(numberofbooks>2){

cout<<"NO MORE BOOKS CAN BE ISSUED!\n";

return 0;

}

return 1;

}

friend class studentlist;

};

class faculty:public member{ //inherited from member

public:

faculty(){

numberofbooks=0;

}

int checkIssue(){ //checks if anymore issue is possible

if(numberofbooks>10){

cout<<"NO MORE BOOKS CAN BE ISSUED!\n";

return 0;

}

return 1;

}

friend class facultylist;

};

//stores transaction details

class transaction{

char date[11];

int memberid;

int bookid;

int isreturned;

public:

void getInfo(int nb, book L[]){ //gets transaction details

cout<<"Enter Transaction Details:\n";

cout<<"Member-Id: ";

cin>>memberid;

cout<<"Book-Id: ";

cin>>bookid;

int f=0;

for(int i=0;i<nb;i++){ //to check if book exists or not, displays error if doesn't

if(bookid==L[i].returnid()){

if(L[i].returnserial()<1){

f=1;

break;

}

}

}

if(f==1){

cout<<"Book Not Available!\n";

return;

}

cout<<"Date: ";

cin>>date;

isreturned=0;

}

int returnbookid(){

return bookid;

}

int memid(){ //returns member id of the user

return memberid;

}

};

class booklist{

static int noofbooks;

book list[100];

public:

void addBook(){

if(nob>100){

cout<<"Book Limit Reached! Contact the Librarian.\n";

return;

}

int f=0;

list[nob].getInfo(nob);

nob++;

cout<<"Book Added!\n";

}

void addexisting(int id){ //updates stock of existing book

if(id>nob){

cout<<"INVALID ID!\n";

return;

}

int qnty;

cout<<"Enter Quantity to Stock Up: ";

cin>>qnty;

list[id].serial=list[id].serial+qnty;

cout<<"Stocked Up!\n";

}

void showBook(int i){

cout<<"SERIAL: "<<list[i].serial;

}

book\* returnbooklist(){

return list;

}

int returnnob(){ //returns number of books

return nob;

}

void checkAvailable(int id){ //check if the book is in stock, displays appropriate message

int flag=0;

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

if(list[i].bookid==id){

if(list[id].serial<1){

cout<<"Book Not Available! Kindly Re-stock.\n";

flag=1;

break;

}

else{

cout<<"Book Available! Copies Remmaining: "<<list[id].serial;

flag=1;

break;

}

}

}

if(!flag)

cout<<"Invalid Book-Id!\n";

}

void updatebooklist(int id, int isreturn){ //updates book stock

if(isreturn)

list[id].serial++;

else

list[id].serial--;

}

friend class book;

};

int booklist::noofbooks = 0;

//stores list of student objects

class studentlist{

static int noofstudents;

student list[100];

public:

void addStudent(){

if(nos>100){

cout<<"Student Limit Reached! Contact the Librarian.\n";

return;

}

list[nos].getInfo();

list[nos].memberid=nos+1;

cout<<"Student Member Id: "<<list[nos].memberid<<endl;

nos++;

cout<<"Student Added!\n";

}

student\* returnstudentlist(){

return list;

}

void showborrowed(int id){ //displays borrow history of student

for(int i=0;i<list[id].numberofbooks;i++){

cout<<"BOOK-ID: "<<list[id].borrowed[i]<<endl;

}

}

static int returnstudents(){

return noofstudents;

}

};

int studentlist::noofstudents=0;

//stores list of faculty objects

class facultylist{

static int nooffaculties;

faculty list[100];

public:

void addFaculty(){

if(nof>100){

cout<<"Faculty Limit Reached! Contact the Librarian.\n";

return;

}

list[nof].getInfo();

list[nof].memberid=nof+1;

cout<<"Faculty Member Id: "<<list[nof].memberid<<endl;

nof++;

cout<<"Faculty Added!\n";

}

faculty\* returnfacultylist(){

return list;

}

void showborrowed(int id){ //displays borrow history of faculty

for(int i=0;i<list[id].numberofbooks;i++){

cout<<"BOOK-ID: "<<list[id].borrowed[i]<<endl;

}

}

static int returnfaculties(){

return nooffaculties;

}

};

int facultylist::nooffaculties=0;

//stores list of transaction objects

class transactionlist{

static int nooftransactions;

transaction list[100];

int canborrow;

public:

int addTransaction(booklist ob){

if(notra>100){

cout<<"Transaction Limit Reached! Contact the Librarian.\n";

return -1;

}

int n=ob.returnnob();

book \*l;

l=ob.returnbooklist();

list[notra].getInfo(n, l);

notra++;

return (notra-1);

}

/ /issues book to student

void studentissued(booklist ob, book L[], int n, int transNo, student l[]){

int k=list[transNo].returnbookid();

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

if(L[i].returnid()==k){

int j=list[transNo].memid();

if(l[j].numberofbooks<2 && L[k].serial>=1){

l[j].borrowed[l[j].numberofbooks]=k;

l[j].numberofbooks++;

L[k].serial--;

}

else{

cout<<"MAX ISSUE LIMIT REACHED!\n";

return;

}

}

}

cout<<"Book Issued to Student!\n";

}

//returns book from student

void studentreturned(booklist ob, book L[], int n, int transNo, student l[]){

int k = list[transNo].returnbookid();

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

if(L[i].returnid()==k){

int j=list[transNo].memid();

if(l[j].numberofbooks<1){

cout<<"No Books to Return!\n";

return;

}

l[j].numberofbooks--;

L[k].serial++;

}

}

cout<<"Book Returned Successfully!\n";

}

//issues book to faculty

void facultyissued(booklist ob, book L[], int n, int transNo, faculty l[]){

int k = list[transNo].returnbookid();

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

if(L[i].returnid()==k){

int j=list[transNo].memid();

if(l[j].numberofbooks<10){

l[j].borrowed[l[j].numberofbooks] = k;

l[j].numberofbooks++;

L[k].serial--;

}

else{

cout<<"MAX ISSUE LIMIT REACHED!\n";

return;

}

}

}

cout<<"Book Issued to Faculty!\n";

}

//returns book from faculty

void facultyreturned(booklist ob, book L[], int n, int transNo, faculty l[]){

int k = list[transNo].returnbookid();

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

if(L[i].returnid()==k){

int j=list[transNo].memid();

if(l[j].numberofbooks<1){

cout<<"No Books to Return!\n";

return;

}

l[j].numberofbooks--;

L[k].serial++;

}

}

// ob.updatebooklist(list[transNo].returnbookid(), 1);

cout<<"Book Returned Successfully!\n";

}

};

int transactionlist::nooftransactions=0;

int main(){

int c,n,transno,avl, bookflag=0, memberflag=0;

int ch;

book \*l; //pointer to book object

//objects for respecive lists

booklist bl;

studentlist sl;

facultylist fl;

transactionlist tl;

//menu

do{

cout<<"\n\n\*\*\*LIBRARY MANAGEMENT SYSTEM\*\*\*\n\n";

cout<<"1. Add New Book.\n";

cout<<"2. Stock up Existing Book.\n";

cout<<"3. Add Student.\n";

cout<<"4. Add Faculty.\n";

cout<<"5. Issue Book.\n";

cout<<"6. Return Book.\n";

cout<<"7. Check Book Availability.\n";

cout<<"8. Show 'Borrow History' of Member.\n";

cout<<"9. Exit.\n";

cout<<"\nEnter Your Choice: ";

cin>>ch;

switch(ch){

case 1:

bl.addBook();

bookflag++;

break;

case 2:

cout<<"Enter Book-Id: ";

cin>>c;

bl.addexisting(c);

break;

case 3:

memberflag=1;

sl.addStudent();

break;

case 4:

memberflag=-1;

fl.addFaculty();

break;

case 5:

if(bookflag==0){

cout<<"NO BOOKS ADDED!!\n";

break;

}

transno = tl.addTransaction(bl);

n=bl.returnnob();

l=bl.returnbooklist();

cout<<"\n1. Issue For Student.\n";

cout<<"2. Issue For Faculty.\n";

cout<<"Enter Your Choice: ";

cin>>c;

//menu for issue

switch(c){

case 1:

student \*LS;

LS = sl.returnstudentlist();

tl.studentissued(bl, l, n, transno, LS);

break;

case 2:

faculty \*LF;

LF = fl.returnfacultylist();

tl.facultyissued(bl, l, n, transno, LF);

break;

}

break;

case 6:

if(bookflag==0){

cout<<"NO BOOKS ADDED!!\n";

break;

}

transno = tl.addTransaction(bl);

n=bl.returnnob();

l=bl.returnbooklist();

cout<<"1. Return For Student.\n";

cout<<"2. Return For Faculty.\n";

cout<<"Enter Your Choice: ";

cin>>c;

//menu for return

switch(c){

case 1:

student \*LS;

LS = sl.returnstudentlist();

tl.studentreturned(bl, l, n, transno, LS);

break;

case 2:

faculty \*LF;

LF = fl.returnfacultylist();

tl.facultyreturned(bl, l, n, transno, LF);

break;

}

break;

case 7:

cout<<"\nEnter Book-Id to be checked: ";

int id;

cin>>id;

bl.checkAvailable(id);

break;

case 8:

if(memberflag==0){

cout<<"NO MEMBERS ADDED!!\n";

break;

}

cout<<"Student(1) or Faculty(0): ";

int k;

cin>>k;

cout<<"Enter Member-Id: ";

int memid;

cin>>memid;

//checking if ny student / faculty has been entered brore, if not displaying error

if(k==1 && memberflag==1){

sl.showborrowed(memid);

}

else if(k==1 && memberflag==-1){

cout<<"NO STUDENT ADDED!\n";

}

else if(k==0 && memberflag==-1){

fl.showborrowed(memid);

}

else{

cout<<"NO FACULTY ADDED!\n";

}

break;

case 9:

exit(0);

break;

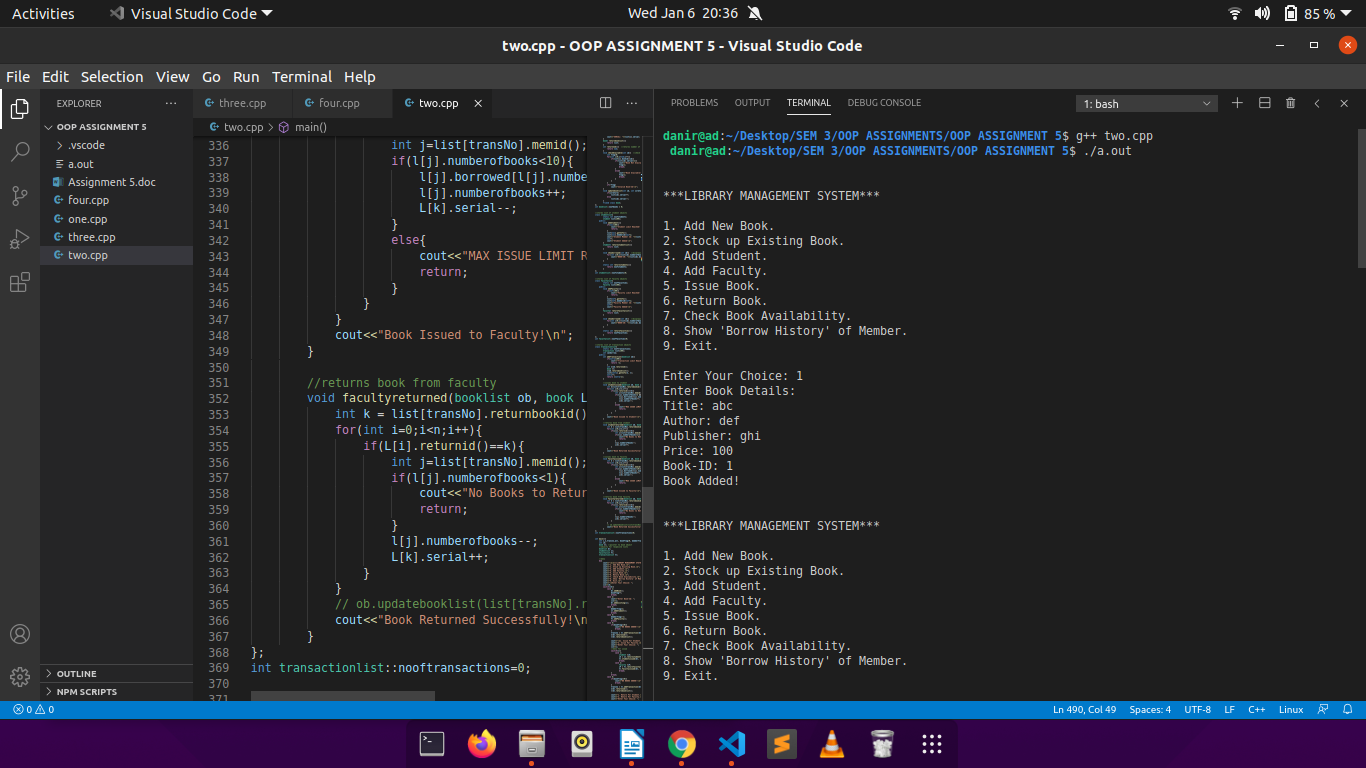
}

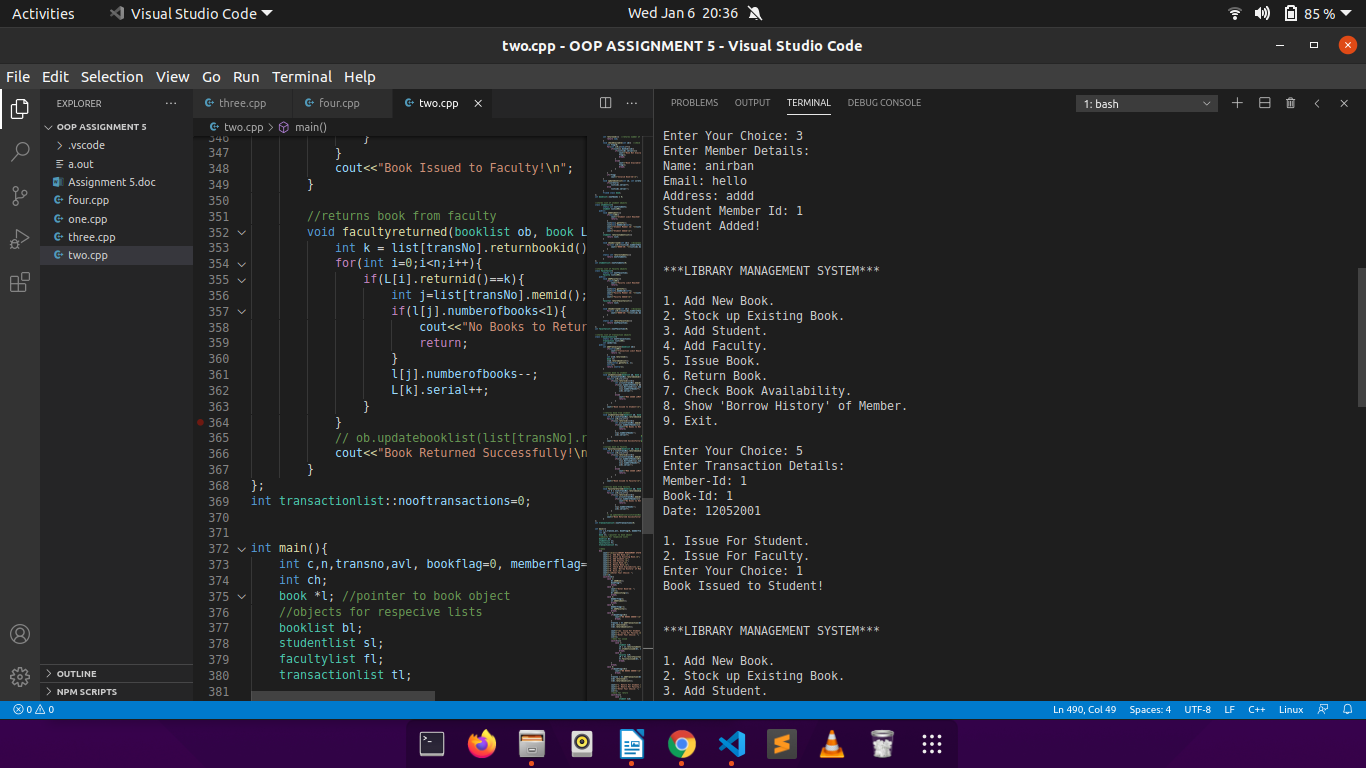
}while(ch!=9);

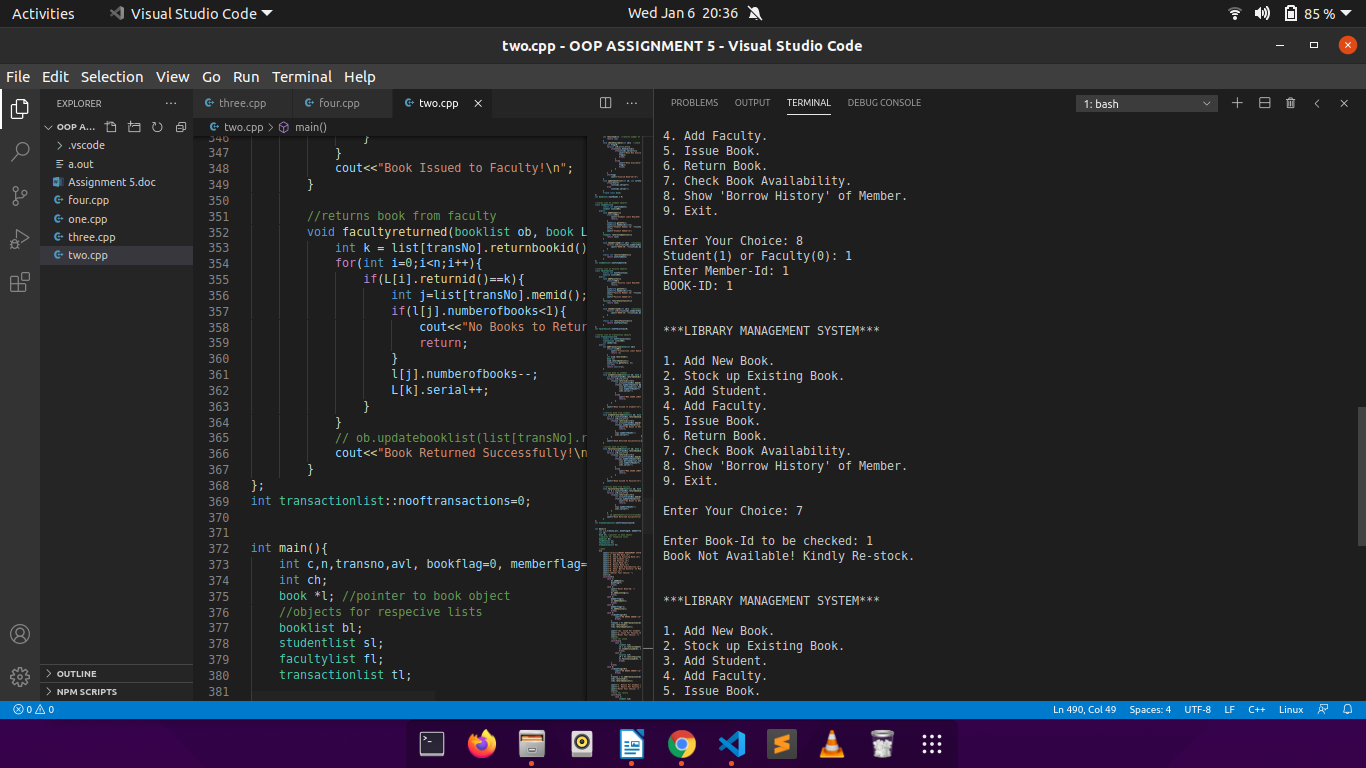
return 0;

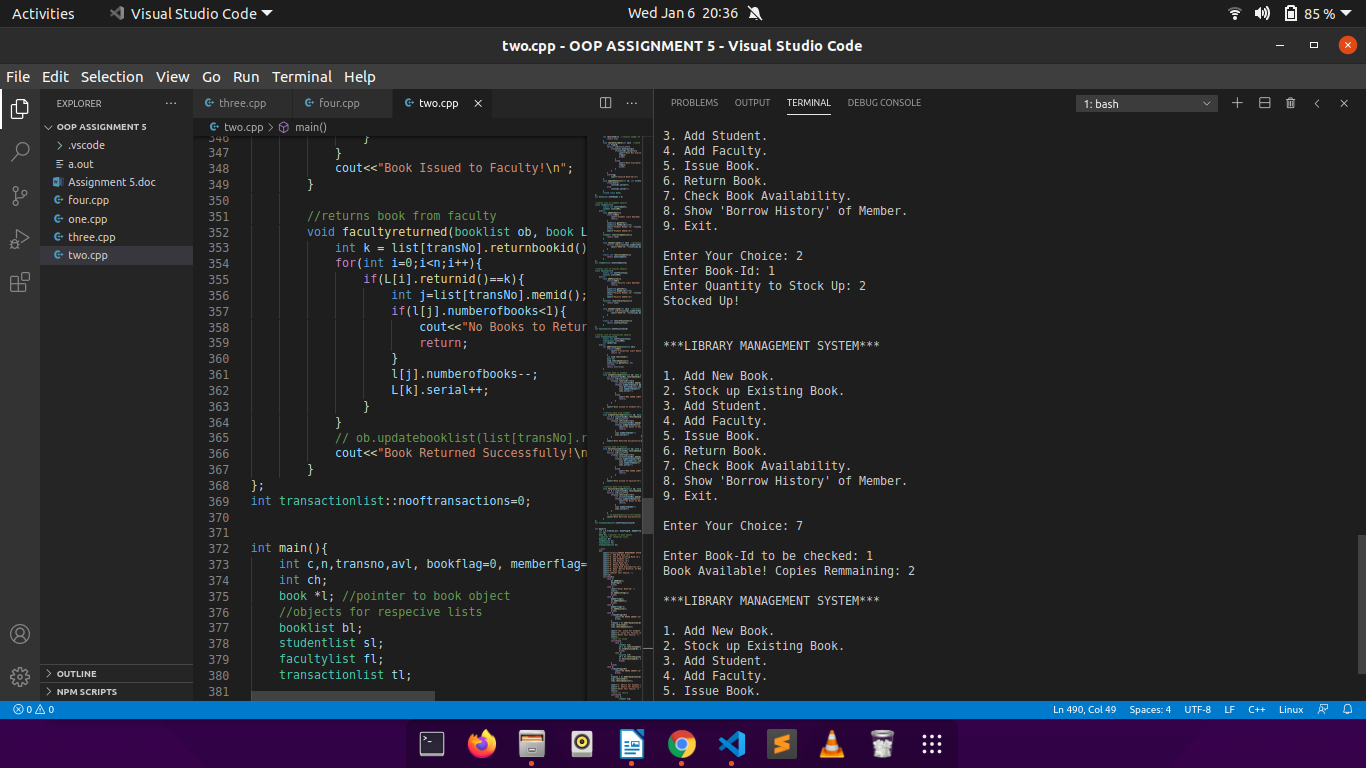
}

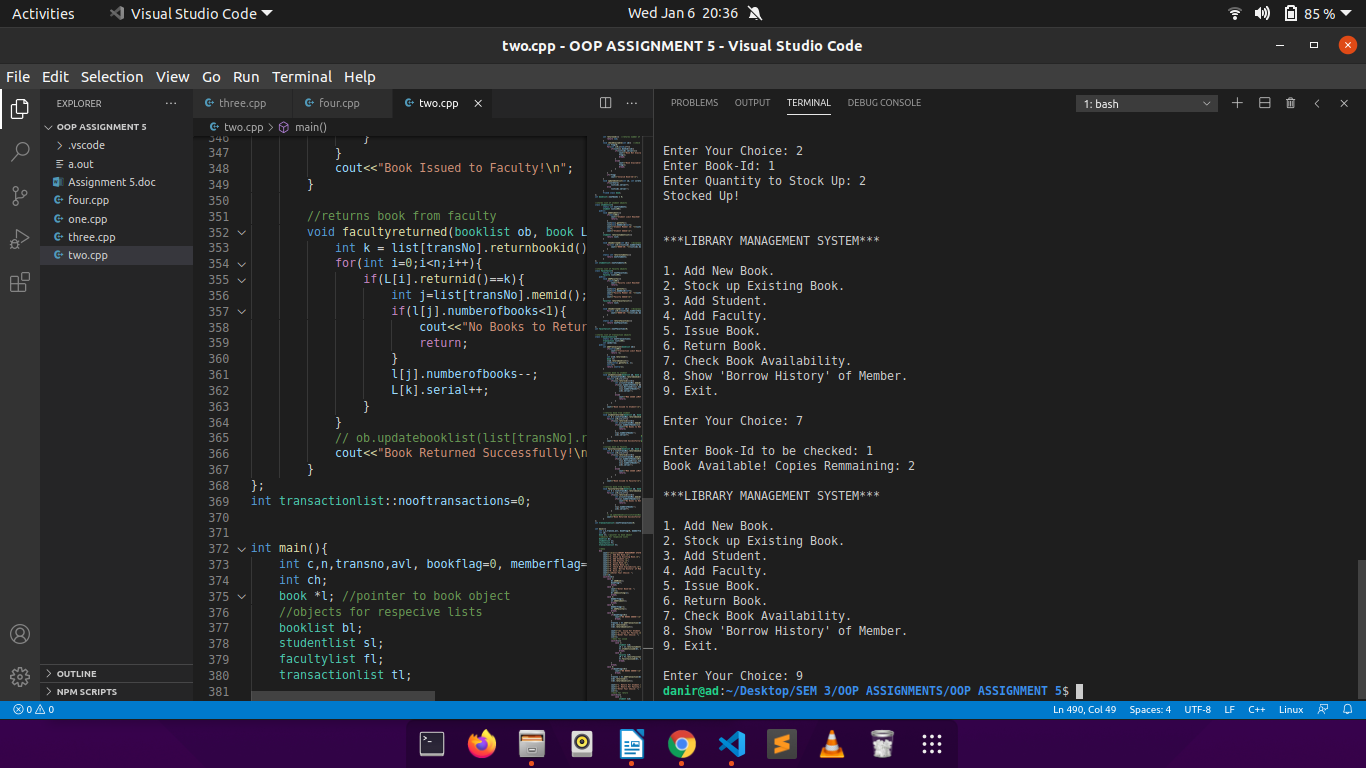
OUTPUT:

****

****







Q3.

CODE:

#include<iostream>

using namespace std;

//stores employee details

class employee{

int emp\_id;

char name[31];

char designation[10];

int basic\_pay;

public:

virtual void getInfo(){

cout<<"Enter Id: ";

cin>>emp\_id;

cout<<"Enter Name: ";

cin>>name;

cout<<"Enter Designation: ";

cin>>designation;

cout<<"Enter Basic Pay: ";

cin>>basic\_pay;

}

virtual float computeSalary(){

return basic\_pay;

}

};

//permanent employees

class permanent:public employee{ //inherits from employee

float salary;

public:

float computeSalary(){ //calculates salary

salary=employee::computeSalary();

float hra = (salary\*0.3);

float da = (salary\*0.8);

salary += da+hra;

return salary;

}

};

//contractual employees

class contractual:public employee{ //inherits from employee

float salary;

float allowance;

public:

void getInfo(){

employee::getInfo();

cout<<"Enter Allowance: ";

cin>>allowance;

}

float computeSalary(){ //calculates salary

salary = employee::computeSalary();

salary += allowance;

return salary;

}

};

int main(){

employee \*ob;

float sal;

ob = new permanent; //permanent employee instance

ob->getInfo();

sal = ob->computeSalary();

cout<<"SALARY= "<<sal<<"\n";

cout<<endl;

ob = new contractual; //contractual employee instance

ob->getInfo();

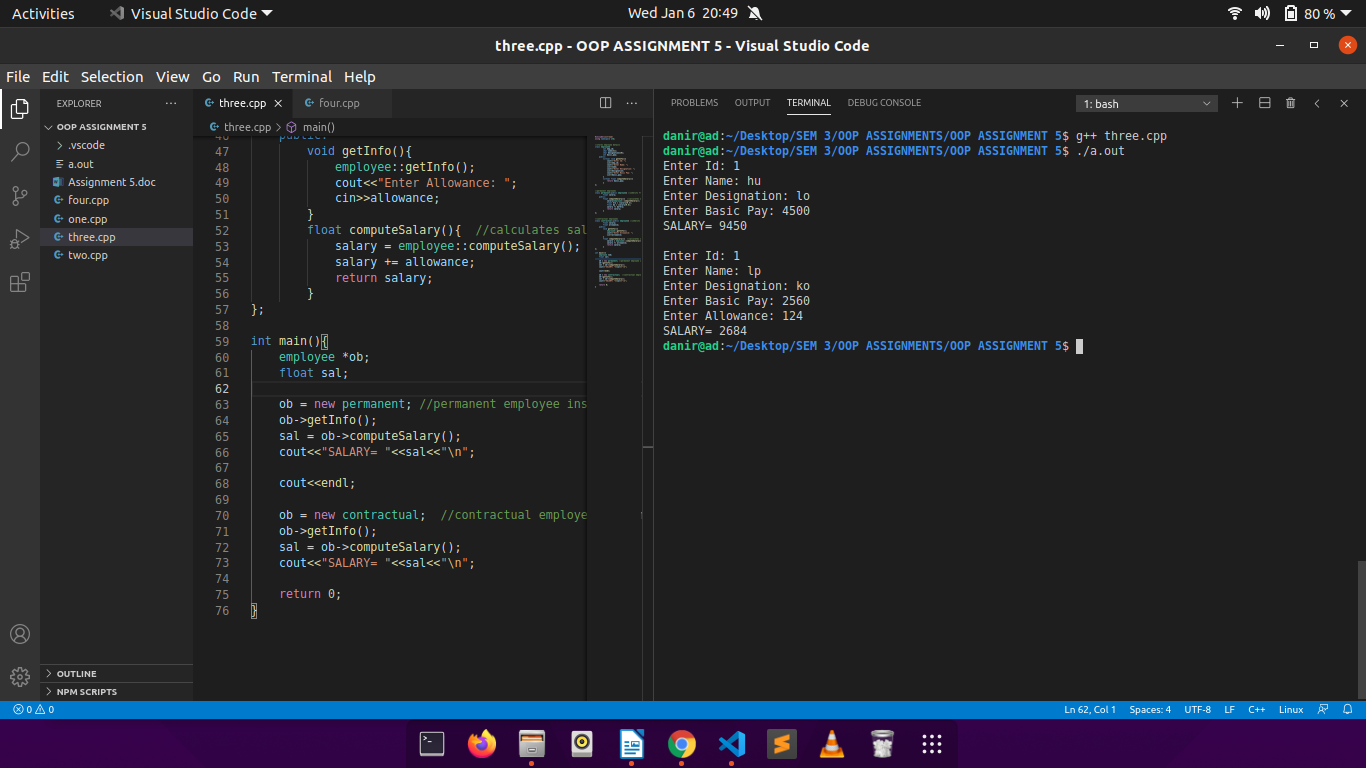
sal = ob->computeSalary();

cout<<"SALARY= "<<sal<<"\n";

return 0;

}

OUTPUT:

****

Q4.

CODE:

#include<iostream>

using namespace std;

//stores cricketer details

class cricketer{

char name[31];

char dob[9];

int matches;

int flag;

public:

cricketer(){

flag=0;

}

virtual void getInfo(){

cout<<"Enter name: ";

cin>>name;

cout<<"Enter dob: ";

cin>>dob;

cout<<"Enter Matches: ";

cin>>matches;

flag=1;

}

virtual void dispInfo(){

cout<<"Name: "<<name<<"\n";

cout<<"DOB: "<<dob<<"\n";

cout<<"Matches: "<<matches<<"\n";

}

int returnMatches(){

return matches;

}

int returnFlag(){

return flag;

}

};

//stores batsman details

class batsman:public virtual cricketer{ //inhetrits from cricketer

int totalRuns;

int avg;

public:

void getInfo(){

if(cricketer::returnFlag()==0)

cricketer::getInfo();

cout<<"Enter Total Runs Scored: ";

cin>>totalRuns;

avg=totalRuns/cricketer::returnMatches();

}

void dispInfo(){

cricketer::dispInfo();

cout<<"Total Runs: "<<totalRuns<<"\n";

cout<<"Average: "<<avg<<"\n";

}

};

//stores bowler details

class bowler:public virtual cricketer{ //inherits from cricketer

int totalWickets;

int runsConceeded;

int oversBowled;

int avg;

public:

void getInfo(){

if(cricketer::returnFlag()==0)

cricketer::getInfo();

cout<<"Enter Runs Connceeded: ";

cin>>runsConceeded;

cout<<"Enter Total Wickets Taken: ";

cin>>totalWickets;

cout<<"Enter Overs Bowled: ";

cin>>oversBowled;

avg=runsConceeded/oversBowled;

}

void dispInfo(){

cricketer::dispInfo();

cout<<"Total Runs Conceeded: "<<runsConceeded<<"\n";

cout<<"Total Wickets Taken: "<<totalWickets<<"\n";

cout<<"Overs Bowled: "<<oversBowled<<"\n";

cout<<"Average: "<<avg<<"\n";

}

};

//stores allrounder details

class allrounder:public batsman, public bowler{ //inherits from both batsman and bowler

public:

void getInfo(){

batsman::getInfo();

bowler::getInfo();

}

void dispInfo(){

batsman::dispInfo();

bowler::dispInfo();

}

};

//stores pair details

class doublewicket\_pair{

batsman bt;

bowler bw;

public:

void get(){

cout<<"Enter Batsman Details:\n";

bt.getInfo();

cout<<"\nEnter Bowler Details:\n";

bw.getInfo();

}

void disp(){

cout<<"BATSMAN DETAILS:\n";

bt.dispInfo();

cout<<"\nBOWLER DETAILS:\n";

bw.dispInfo();

}

};

int main(){

cricketer \*ob;

cout<<"Enter Batsman Details:\n";

ob = new batsman; //batsman instance

ob->getInfo();

cout<<"\n";

ob->dispInfo();

cout<<"\nEnter Bowler Details:\n";

ob = new bowler; //bowler instance

ob->getInfo();

cout<<"\n";

ob->dispInfo();

cout<<"\nEnter All-Rounder Details:\n";

ob = new allrounder; //allrounder instance

ob->getInfo();

cout<<"\n";

ob->dispInfo();

cout<<"\nEnter Pair Details:\n";

doublewicket\_pair p; //pair object

p.get();

cout<<"\n";

p.disp();

return 0;

}

OUTPUT:

