Name : Aritra Das

Roll No: 75

Assignment 6

1.

/\*In a library, for each book book-id, serial number (denotes copy number of a

book), title, author, publisher and price are stored. Book-id and serial number

together will be unique identifier for a book. Members are either student or

faculty. Each member has unique member-id. Name, e-mail, address are also to

be stored. For any transaction (book issue or return), members are supposed to

place transactions slip. User will submit member-id, book-id, and serial number

(only for book return). While processing a transaction, check the validity of the

member. While issuing, availability of a copy of the book is to be checked. While

returning a book, it is to be checked whether this copy was issued to the

member or not. A student member can have 2 books issued at a point of time.

For faculty members it is 10. Transaction information is to be stored like date of

transaction, member-id, book-id, serial number, returned or not. An entry is

made when book is issued and updated when the book is returned. For storing

the information consider files.

Design the classes and implement.\*/

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

#include<fstream>

using namespace std;

class Book{

    public:

    int b\_id,s\_no,price;

    string title,author,pub;

    vector<int> m\_list;

    Book \*next;

};

class Member{

    public:

    int m\_id;

    string name,email;

    int count;

    char type;

    vector<int> b\_list;

    Member \*next;

};

class BookList{

    Book \*head;

    public:

    BookList(){

        head = NULL;

    }

    void push(int x, int t,string y,string z, string p){

        Book \*temp = new Book;

        temp->b\_id = x;

        temp->s\_no = 1;

        temp->title = y;

        temp->author = z;

        temp->pub = p;

        temp->price = t;

        temp ->next = NULL;

        if(head == NULL)

        head = temp;

        else{

            Book\* temp1;

            temp1 = head;

            while(temp1->next=NULL){

                if(!checkUnique(x)){

                    cout<<"Book ID already exists, increasing the Serial Count\n";

                    temp1->s\_no++;

                    return;

                }

                temp1=temp1->next;

            }

            temp1->next = temp;

        }

    }

    void display(){

        Book \*temp1;

        temp1= head;

        while(temp1!=NULL){

            cout<<"Book ID : "<<temp1->b\_id<<" Title: "<<temp1->title<<" Author: "<<temp1->author<<" Publisher: "<<temp1->pub<<" Serial No: "<<temp1->s\_no<<" Price: "<<temp1->price<<endl;

            temp1=temp1->next;

        }

    }

    bool checkUnique(int acNo1){

        Book\* temp;

        temp = head;

        if(head==NULL)

        return true;

        while(temp!=NULL){

            if(temp->b\_id == acNo1)

            return false;

            temp = temp->next;

        }

        return true;

    }

    void add\_M\_List(int x,int y){

        Book\* temp;

        temp = head;

        while(temp!=NULL&&temp->b\_id != y){

            temp = temp->next;

        }

        temp->m\_list.push\_back(x);

    }

    void remove\_M\_List(int x,int y){

        Book\* temp;

        temp = head;

        while(temp!=NULL&&temp->b\_id != y){

            temp = temp->next;

        }

        for (auto i =temp->m\_list.begin();  i != temp->m\_list.end(); ++i){

            if(\*i==x){

                temp->m\_list.erase(i);

                break;

            }

        }

    }

    void inc\_count(int x){

        Book\* temp;

        temp = head;

        while(temp!=NULL&&temp->b\_id != x){

            temp = temp->next;

        }

        temp->s\_no++;

    }

    void dec\_count(int x){

         Book\* temp;

        temp = head;

        while(temp!=NULL&&temp->b\_id != x){

            temp = temp->next;

        }

        if(temp->s\_no>0)

        temp->s\_no--;

        else

        cout<<"Insufficient Books\n";

    }

};

class MemList{

    Member \*head;

    public:

    MemList(){

        head = NULL;

    }

    void push(int x,string y,string z,char t){

        Member \*temp = new Member;

        temp->m\_id = x;

        temp->email = z;

        temp->name = y;

        temp->type = t;

        temp->count = 0;

        temp ->next = NULL;

        if(head == NULL)

        head = temp;

        else{

            Member\* temp1;

            temp1 = head;

            while(temp1->next=NULL){

                if(!checkUnique(x)){

                    cout<<"Member already exists\n";

                    return;

                }

                temp1=temp1->next;

            }

            temp1->next = temp;

        }

    }

    void display(){

        Member \*temp1;

        temp1= head;

        while(temp1!=NULL){

            cout<<"Member ID : "<<temp1->m\_id<<" Email: "<<temp1->email<<" Name: "<<temp1->name<<endl;

            temp1=temp1->next;

        }

    }

    bool checkUnique(int acNo1){

        Member\* temp;

        temp = head;

        if(head==NULL)

        return true;

        while(temp!=NULL){

            if(temp->m\_id == acNo1)

            return false;

            temp = temp->next;

        }

        return true;

    }

    void add\_B\_List(int y,int x){

        Member\* temp;

        temp = head;

        while(temp!=NULL&&temp->m\_id != y){

            temp = temp->next;

        }

        if(temp->type == 's' && temp->count < 3)

        temp->b\_list.push\_back(x);

        else if(temp->type == 'f' && temp->count < 8)

        temp->b\_list.push\_back(x);

        else

        cout<<"Limit exceeded\n";

    }

    void remove\_B\_List(int y,int x){

        Member\* temp;

        temp = head;

        while(temp!=NULL&&temp->m\_id != y){

            temp = temp->next;

        }

        for (auto i =temp->b\_list.begin();  i != temp->b\_list.end(); ++i){

            if(\*i==x){

                temp->b\_list.erase(i);

                break;

            }

        }

    }

};

class Transaction{

    public:

    BookList a;

    MemList a1;

    fstream fio;

    void addB(){

        fio.open("books.txt",ios::out | ios::app);

        int b,c;

        string d,e,f;

        cout<<"Enter Book ID, Title, Author, Publisher, Price:\n";

        cin>>b>>d>>f>>e>>c;

        a.push(b,c,d,e,f);

        fio<<b<<d<<f<<e<<c<<endl;

        fio.close();

    }

    void viewB(){

        fio.open("books.txt",ios::in);

        cout<<"The list of books are:\n";

        a.display();

        fio.close();

    }

    void addS(){

        fio.open("members.txt",ios::out | ios::app);

        int b;

        string d,e;

        cout<<"Enter Member ID, Name, Email:\n";

        cin>>b>>d>>e;

        fio<<b<<d<<e<<endl;

        a1.push(b,d,e,'s');

        fio.close();

    }

    void addM(){

        fio.open("members.txt",ios::out | ios::app);

        int b;

        string d,e;

        cout<<"Enter Member ID, Name, Email:\n";

        cin>>b>>d>>e;

        fio<<b<<d<<e<<endl;

        a1.push(b,d,e,'f');

        fio.close();

    }

    void viewM(){

        fio.open("members.txt",ios::in);

        cout<<"The list of Members are:\n";

        a1.display();

        fio.close();

    }

    void issue(){

        fio.open("transactions.txt",ios::out|ios::app);

        cout<<"Available Books:\n";

        a.display();

        int k,l;

        cout<<"Enter Book ID and Member ID\n";

        cin>>k>>l;

        fio<<"Book ID: "<<k<<" Member ID"<<l<<" Status: Issued"<<endl;

        a1.add\_B\_List(k,l);

        a.add\_M\_List(l,k);

        a.dec\_count(k);

        cout<<"Book Issued\n";

        fio.close();

    }

    void retur(){

        fio.open("transactions.txt",ios::out|ios::app);

        int k,l;

        cout<<"Enter Book ID and Member ID\n";

        cin>>k>>l;

        fio<<"Book ID: "<<k<<" Member ID"<<l<<" Status: Returned"<<endl;

        a1.remove\_B\_List(k,l);

        a.remove\_M\_List(l,k);

        a.inc\_count(k);

        cout<<"Book Returned\n";

        fio.close();

    }

};

int main(){

    Transaction t;

    for(;;){

        cout<<"Enter:\n1. Add a Book\n2. Add a Student\n3. Add a Faculty\n4. Issue a Book\n5. Return a Book\n6. Show all Books\n7. Show all Members\n8. Exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                t.addB();

                break;

            }

            case 2:{

                t.addS();

                break;

            }

            case 3:{

                t.addM();

                break;

            }

            case 4:{

                t.issue();

                break;

            }

            case 5:{

                t.retur();

                break;

            }

            case 6:{

                t.viewB();

                break;

            }

            case 7:{

                t.viewM();

                break;

            }

            case 8:{

                cout<<"Exiting ... \n";

                exit(0);

            }

            default:{

                cout<<"Wrong choice\n";

                break;

            }

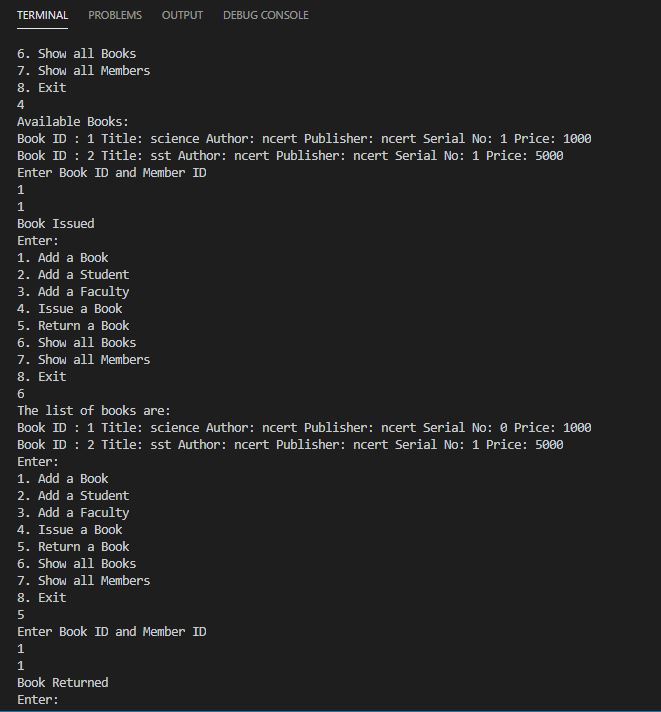
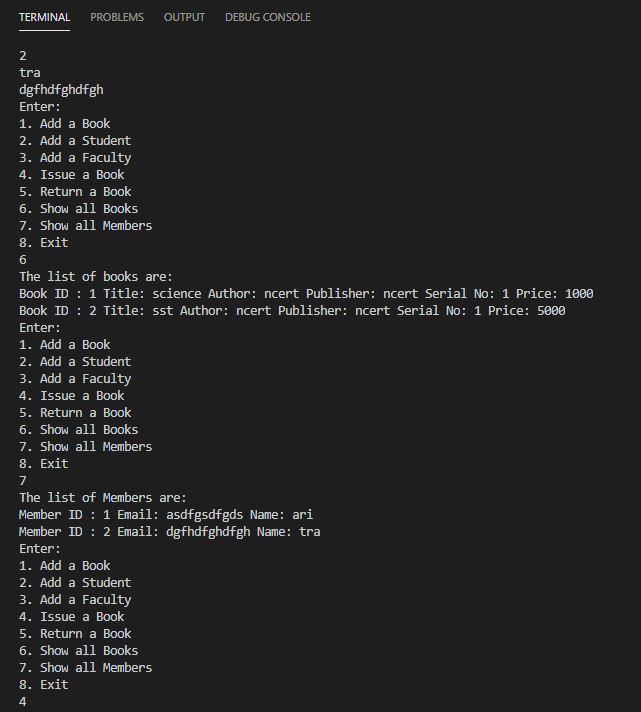
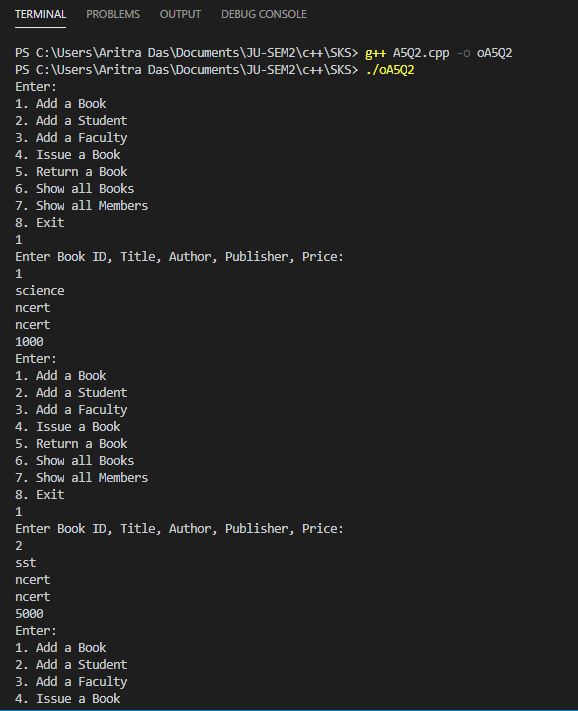
        }

    }

    return 0;

}

Output:



2.

/\*

Consider a class Student with roll, name and score as attributes. Support to take

and display data is also there. One wants to works with array of Student

objects. May collect data for array elements, display those. In case index goes

out of bounds, exception is to be raised with suitable message.

\*/

#include<iostream>

#include<string>

using namespace std;

class Student{

    int roll,score;

    string name;

    public:

    Student(){

        roll = 0;

        score = 0;

        name = "";

    }

    void input(){

        cout<<"Enter Roll, Name and Score\n";

        cin>>roll>>name>>score;

    }

    void display(){

        cout<<"Roll: "<<roll<<" | Name: "<<name<<" | Score: "<<score<<endl;

    }

};

int main(){

    for(;;){

        strt:

        cout<<"Enter :\n1. For array of objects initialization and operations\n2. To Exit\n";

        int c1,c2;

        cin>>c1;

        switch(c1){

            case 1:{

                Student \*arr;

                cout<<"Enter length of array of objects:\n";

                int x;

                cin>>x;

                arr = new Student[x];

                for(;;){

                    cout<<"Enter :\n1. For taking input of all elements in array of objects\n2. For displaying i'th object in array of objects\n3. For displaying all elements of array of objects\n4. To go back to main menu\n";

                    cin>>c2;

                    switch(c2){

                        case 1:{

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

                            arr[i].input();

                            break;

                        }

                        case 2:{

                            cout<<"Enter index:\n";

                            int y;

                            cin>>y;

                            if(y>x||y<0){

                                cout<<"Out of bounds, invalid position\n";

                                break;

                            }

                            else{

                                cout<<"The "<<y<<"'th object in array:\n";

                                arr[y].display();

                                break;

                            }

                            break;

                        }

                        case 3:{

                            cout<<"All elements in array of objects: \n";

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

                                cout<<i+1<<" . ";

                                arr[i].display();

                            }

                            break;

                        }

                        case 4:{

                            goto strt;

                        }

                        default:{

                            cout<<"Wrong Input";

                            break;

                        }

                    }

                }

                break;

            }

            case 2:{

                cout<<"Exiting ... ";

                exit(0);

            }

            default:{

                cout<<"Wrong Input";

                break;

            }

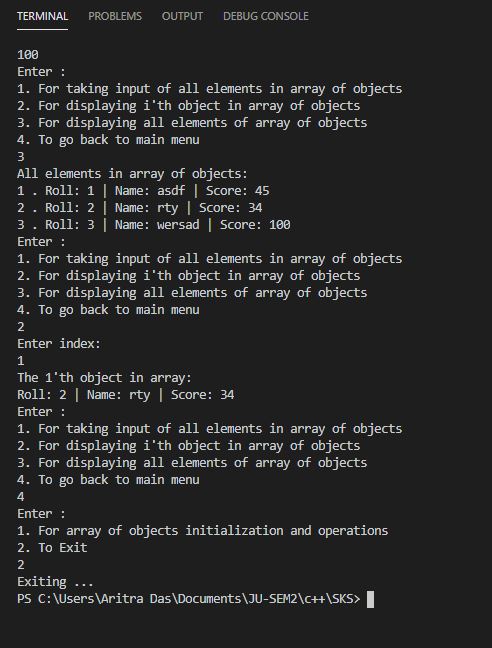
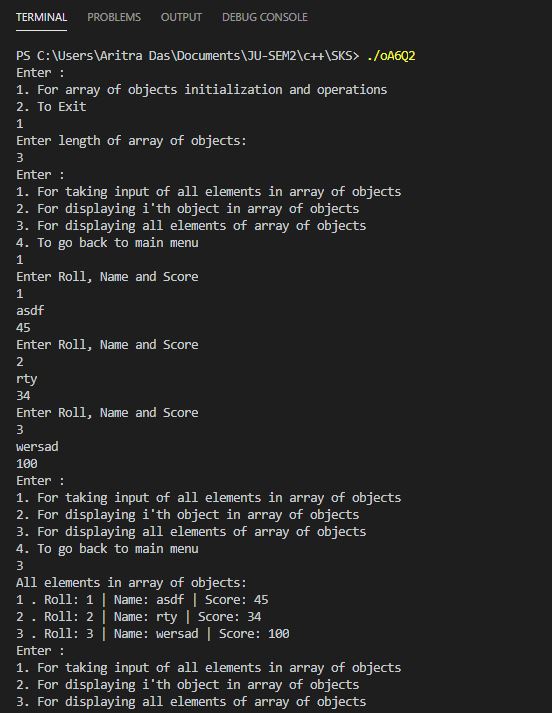
        }

    }

    return 0;

}

Output:



3.

/\*

Implement a class template for 1D array. Elements may be any basic data type.

Provision to find maximum element, sum of the elements must be there.

\*/

#include<iostream>

using namespace std;

class ARRAY{

    private:

    int \*arr;

    int size;

    public:

    ARRAY(int n){

        size = n;

        arr = new int[n];

        for(int i=0;i<n;i++){

            cout<<"Enter element "<<i+1<<" : \n";

            cin>>arr[i];

        }

    }

    int& operator[](int i){

        return arr[i];

    }

    int max(){

        int max=0;

        for(int i=0;i<size;i++){

            if(arr[i]>max)

            max=arr[i];

        }

        return max;

    }

    int sum(){

        int sum = 0;

        for(int i=0;i<size;i++)

        sum+=arr[i];

        return sum;

    }

};

int main(){

    for(;;){

        strt:

        cout<<"Enter :\n1. For new Array and operations\n2. To Exit\n";

        int c1,c2;

        cin>>c1;

        switch(c1){

            case 1:{

                cout<<"Enter length of new Array:\n";

                int x;

                cin>>x;

                ARRAY t(x);

                for(;;){

                    cout<<"Enter :\n2. For getting i'th element\n3. For getting sum of array\n4. For getting max of array\n5. To go back to main menu\n";

                    cin>>c2;

                    switch(c2){

                        case 2:{

                            cout<<"Enter index:\n";

                            int y;

                            cin>>y;

                            if(y>x||y<0){

                                cout<<"Out of bounds, invalid position\n";

                                break;

                            }

                            else{

                                cout<<"The "<<y<<"'th element is : "<<t[y]<<endl;

                                break;

                            }

                        }

                        case 3:{

                            cout<<"The sum of array is: "<<t.sum()<<endl;

                            break;

                        }

                        case 4:{

                            cout<<"The max element of array is: "<<t.max()<<endl;

                            break;

                        }

                        case 5:{

                            goto strt;

                        }

                        default:{

                            cout<<"Wrong Input";

                            break;

                        }

                    }

                }

                break;

            }

            case 2:{

                cout<<"Exiting ... ";

                exit(0);

            }

            default:{

                cout<<"Wrong Input";

                break;

            }

        }

    }

    return 0;

}

Output:

