**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: B.Tech. Sem III

**Subject: Data Structures Division: E**

**Mini - Project Report**

|  |  |  |
| --- | --- | --- |
| Name of the Project: | **BOOK RENTING PORTAL** | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| E1 | E016 | Vaishnavi Rathod |
| E3 | E047 | Riya Tendulkar |
|  |  |  |
| Date of Submission: 02/10/2019 | | |
| Grades/Marks | | |

A.1 **Aim of the Project:**

The aim of our project is to implement some of the many data structures we gained knowledge about to solve some real-life problems.

*What exactly is our project?*

Our Project is a Book Renting Portal. Sites like amazon, flipkart, snapdeal usually sell books from the publisher to the customer and act as a medium between them. We aim to do the same but without involving the buying and selling of the books. Buying and selling often gets expensive and unnecessary for people who want to read the book once and do not want to spend so much money in actually buying the book. So, in our project customers can rent books from users who are ready to give their books on rent. This is a good deal for both the seller (person giving his book on rent) and buyer (person renting the book) as the seller retains the books and earns money and the buyer gets to read his desired book without buying a book.

*How does our project work?*

We have predefined database of customers (name, phone number, address, locality) and the books (name, genre, price) they are selling.

Any user or new customer can go to the portal and enter the genre of the book he wants to rent. He will be shown a bunch of books of the selected genre from the database and he can add the books which he wants to rent to the cart.

After adding all the desired books to the cart, he will be taken to the check-out counter.

As there is no third-party seller here, the buyer and seller need to meet and exchange books. To make this task easier for both of them, we find the most optimal and shortest path the buyer will have to take to reach to the seller. We tell the seller the distance and path he will have to travel to reach the buyer and we finalize the order.

Here is a representation of the flow of our project:

Start

User comes to the portal

He logins or enters his details

He enters the genre of the book he wants to read

Books of the selected genre are displayed from the database

User can add the desired books to the cart

User is asked if he wants to try another genre

User is taken to the cart where he can perform various operations

Check-Out Counter:

User is given a summary of his cart, total bill, the shortest distance and path to each user whose book is being rented

End

A.2 **Application of the Project:**

Use of libraries is very popular all over the world. That is the very idea that inspired us to create this project using various data structures. We also take some amount of inspiration from online retailers like amazon. We aim at bringing out this project to the world that will help connect people over their love for books. You can rent your book to others and earn some money or decide to rent a book you wish to read from the person available nearest to you and save some money than actually buying the book.

A.3 **Data Structures used and justification:**

In our project we are using 3 data structures: **Arrays, Linked List and Graphs**

**ARRAYS**

Arrays are an extremely used data structure of our project and is used almost everywhere. Arrays are easily accessible and can store a large amount of data and so were very crucial in our project. The reasons for using arrays and where they are used are as follows:

|  |  |
| --- | --- |
| Use of Array | Reason |
| Storing two databases: Customers and Books | There were many fields of the customers and books to be stored and so multidimensional arrays were used as they are convenient for storing large data and displaying the data becomes easy as well |
| Storing adjacency matrix of the graph | We have used Dijkstra’s Algorithm in our project and for that we needed an adjacency matrix of the graph. A multi-dimensional graph was used to store the distances (adjacency matrix) |
| Dijkstra’s Algorithm Table | For implementing Dijkstra’s algorithm, we usually make a table and implement it so for converting the algorithm in code form, we have transformed that table in the form of a multidimensional matrix |
| Storing Cities | The cities of the distances we have used a single dimensional array. This is connected to the adjacency matrix of the graph |

**LINKED LIST**

We have majorly used linked list in 2 parts of our project-**Cart and Displaying path of Dijkstra’s Algorithm**.

*Linked List in Cart:*

* **Heterogenous Data-Type**

As our books have many features- Id, title, price, genre, customer code- and many of them have different data-types, using linked list was efficient as linked lists support heterogenous data type.

* **Dynamic Nature**

Unlike arrays, linked lists are dynamic, that is they can grow and shrink at run-time. As the number of items the user will be adding to the cart is variable and dynamic we need to go on adding things to the previous cart and so as items are added to the cart, nodes are added to the linked list.

* **Insertion and Deletion**

Insertion and deletion is easy in linked list. The main functionality of any cart is adding and deleting items. Using linked list there is no restriction as to where an item is added or deleted from. You can delete any item irrespective of where it is in the linked list.

* **Functionalities**

We have implemented the following functionalities of a linked list (idk if this to be added in applications):

* Insert node at beginning
* Remove node depending on the data
* Remove node depending upon the serial no.
* Count nodes of the list
* Display Linked List
* Delete entire list
* Sort List
* Delete Duplicates in an unsorted list
* Sort and bring together the nodes having same parameter(genre)

*Linked List in Graph:*

In our project, displaying the shortest distance between two paths is a major part. We are doing this using Dijkstra’s Algorithm (Please refer down for Dijkstra’s Matrix representation). At the end while displaying the path we need to traverse through the last column of the matrix which stores the parent. Now doing this using arrays is not possible as the number of parents of each vertex varies and is dynamic. For example, a vertex maybe one vertex away from the parent (Example: source-node1-destination) where as another could have multiple vertices in between (Example: source-node1-node2-node3-node4-destination). So we use linked lists for this as linked lists are dynamic in nature and the size is not required to be predefined. We add nodes to the beginning of the list and display the linked list at the end. This displays the path in the proper order.

**GRAPH**

In this project, graph is used to find the shortest distance and path between the seller and buyer.

For this we use Dijkstra’s Algorithm Dijkstra’s Algorithm uses the greedy approach. We provide a weighted graph, source and destination. Using this the algorithm calculates the shortest distance and algorithm. We use a table for the implantation of the algorithm which is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Vertex** | **Visited** | **Distance** | **Parent** |
| Source | 1 | 0 | 0 |
|  |  |  |  |
|  |  |  |  |

Vertex: Vertex for which the shortest distance is calculated

Visited: Keeps a track if the vertex is visited or not

Distance: Holds the shortest distance between the source and vertex

Parent: Holds the vertex who is the parent

**We also implement the two algorithms for displaying the graph-**

**Breadth First Search (BFS) and Depth First Search(DFS)**

A.4 **Description of project:**

Class Diagrams:

**class cartMenu**

* countNodes()
* add()
* deleteEntire()
* deleteItem()
* deleteOnBookid()
* deletDuplicates()
* display()
* totalAmount()
* sortAscending()
* sortDescending()
* connectCustomer()
* getDistance()
* menu()

**class Graph**

* setGraph()
* dijkstra()
* BFS()

**Functions of the Code**

* displayCustomers()
* displayGraph()
* displayBooks()
* loginAccount()
* showBooksMenu()
* DFS()
* admin()
* main()

Here are the function names and their descriptions:

* **void displayCustomers()**

Since we have a hard coded database for the previously registered customers, we use this function to access that multidimensional array and print all the details present in the database .We print id, name, number, username, password, address and locality.

* **void displayGraph**()

We access the graph that has stored all the cities i.e. the places available for renting a book from. We print the adjacency matrix of all the interconnecting distances between these places.

* **void displayBooks()**:

We have a database for the books hard coded. We use this function to print all the details of the book to the user. Details printed are sr no, book code, book name, genre and price.

* **void loginAccount()**

Here we ask the user if he is already registered and retrieve all of his/her information from the customer database. For that the user must correctly enter the username and password. These details are matched with all the previous records. If the user name or password entered is incorrect, the user gets to try again 5 more times. If the user is not registered, he can enter name, locality, address and continue.

* **void showBooksMenu()**

Takes the input from the user as to which genre books he wishes to view and select to rent. It shows the book details to the user and enables him to make a choice of book and add it to cart.

* **void DFS(int)**

One of the major implementations of graph, this function displays the entire graph using ‘Depth First Search Algorithm’. To avoid complexity, we have used the predefined class of Stack for this.

* **void admin()**

This function is called after the customer is done with his order. It is for the admin of the code to perform functions like displaying the databases or graph or changing and editing the graph.

* **int main()**

We use main to call function: loginUser() and after the loginUser() is done we call admin().

**GRAPH CLASS-**

* **void setGraph()**

This function allows the user to create his own graph. In that he can enter the number of vertexes and the edges. The vertexes here are the nodes or number of places and edges are the distances between the nodes. Based on all the information the user enters an adjacency matrix is created.

* **void BFS(int, int[][], string[], int)**

This function prints the graph entered by the user. It is called from the setGraph() function and uses the ‘Breadth First Search’ algorithm.

* **void dijkstra(int [][],string [],string ,string ,int )**

The heart of the project, this function is extremely crucial in the proper functioning of our book renting portal. Dijkstra’s algorithm is implemented here using graphs. All initial values are set of dijkstra's matrix, i.e, visited=0, distance=infinity ,parent=0 ,first.

After assigning the source code, the process of finding shortest distance between the source and destination begins. The shortest path is stored in a linked list. We display the details of the shortest distance and the path to the user.

**CARTMENU CLASS-**

* **int countNodes()**

It will count the number of nodes present in the linked list. In simpler terms,it will count the number of books added in the shopping cart.

* **void add(string , string, string ,string ,int)**

This function adds a new node to the list like adding an element to the cart. A new node is created in the linked list that stores sr no, book code, book name, genre and price. And it gets added to the cart.

* **void deleteItem(int)**

This function deletes the item/node from the cart. The user will enter the serial number of the item which will act as the position of the node(book) and hence it will get removed from the cart.

* **int totalAmount()**

For all the books present in the cart, it calculates the total price of the items present in the cart and displays to the user.

* **void display()**

It prints the items present in the shopping cart. It shows all the details of each and every item in the cart and also the total no of items and total amount.

* **void deleteOnBookid(string)**

If the user enters some id of a book, this function will help him delete that book from the cart.

* **void deleteEntire()**

If the user wants to delete all the linked list ie all books from the shopping cart. This function will allow him to do so.

* **void sortAscending()**

This function will sort the shopping cart in the ascending according to the price of the book in the cart.

* **void sortDescending()**

This function will sort the shopping cart in the descending according to the price of the book in the cart.

* **void connectCustomer()**

This acts as the check out counter of our portal. The details of the book will be displayed to the user. Also, the details of the seller is shown like name, phone number, address and locality. The seller is the person who has the book that the user wants.

* **void deleteDuplicates()**

If a user has accidentally entered double items, this function will delete all the duplicates from the cart.

* **void getDistance(string , string)**

This is a friend function of the graph class as well.

The details of the buyer and seller is passed to this function, their localities act as the source and destination nodes for finding the smallest distance between them to reach each other.

* **void menu()**

It displays the menu available in the cart. It gives the user various functionalities like removing items, sorting them according to the prices, displaying the item details, total no and total amount, deleting items, deleting entire cart.

**So the flow of the project goes as such:**

1. The user logins using his username and password. If he gives an incorrect input he can try for 5 more times.

*loginAccount() called*

1. After logging in, he will enter the genre of the books he wants to buy. He can enter multiple genre and books. After that is done with adding the books to his cart, he will be taken to the cart.

*showBooksMenu() called*

1. In the cart he can perform the multiple functionalities stated above.

*cartMenu class implemented*

1. Then he is taken to the check out counter, where he is shown the details of individual books and it’s seller details.

*connectCustomer() called*

1. Along with this the shortest path and distance from the user to the seller is shown.

*getDistance()* is called which implents class Graph

1. after this we go back to the main function and the admin function is called.

*admin() called*

1. Admin function implements many functions mainly database related functions
2. We go back to main and the program is terminated

Flow of the project:

**admin()**

**main()**

end

**loginAccount()**

**showBooksMenu()**

**menu()**

[Class cartMenu]

**connectCustomer()**

[class cartMenu]

**getDistance()**

[class cartMenu]

**dijkstra()**

[class Graph]

A.5 **Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| E016  E047 | Vaishnavi Rathod  Riya Tendulkar | LoginAccount  Cart  Outputs  Cart  Dijkstra’s Algorithm (Graph)  Commenting & Formatting |

A.6 **Properly commented Software Code:**

/\*WELCOME TO OUR DS PROJECT

This is a book renting portal where users can read books on rent

The Flow of the Code is:

Header Section

Declaration of global variables and structures

class Graph

class cartMenu

displayCustomers()

displayBooks()

displayGraph()

loginAccount()

showBooksmenu()

DFS()

admin()

main()

For a simplified version of our project flow, please refer our detailed reports

Creaters:

Riya Tendulkar and Vaishnavi Rathod\*/

//Header Section

#include<iostream>

#include<cmath>

#include<conio.h>

#include <stdlib.h>

#include <string.h>

#include<iostream>

#include<cstring>

#include<stack>

#include<queue>

using namespace std;

//ProtoType of the functions

void loginAccount(void);

void showBooksMenu(void);

void displayBooks(void);

void displayCustomers(void);

void showBooksMenu(void);

//Declaring the structures used in the code

//This structure is used for printing the shortest path from user to seller using linked list

struct path{

string nextVertex;

struct path \*next;

};

//This structure is used for the cart class using linked list

struct cart{

string title;

string customerId;

string bookId;

int price;

string genre;

struct cart \*next;

};

//Creating databses

/\*Adding customers to the database

Now here we have a matrix with rows=10 and coloumns=7

Here number of rows is the number of customers and number of coloumns is the criterias for database, they are as follows:

0->Customer Id

1->Name

2->Phone Number

3->Username

4->Password

5->Address

6->Locality

\*/

string customers[10][7]={{"1","Diya","4653258765","diyaarora123","diyaroxx","4,Orchard Street","Kandivali"},

{"2","Manoj","1234567890","manishere5","manojissad","12,Thakur Complex","Malad"},

{"3","Sunny","7587465856","sunisfun","sunyybunny","20,Sunrise Building","Thane"},

{"4","Gargi","3569687253","chefgargi","ilovevaish","604,Vasant Vihar","Powai"},

{"5","Pranay","56748937520","kingtopper","hatemeplease","41,Riddhima Circle","Vile Parle"},

{"6","Monica","3434546777","cleanfreak","bestcookever","19,Pride Residency","Worli"},

{"7","Harry","1029384756","guspowered","mypassword9","40,Top Apartments","Mulund"},

{"8","Shivam","2346507876","ohmyfriend","shivbhakt","12,Banaras Galli","Vikhroli"},

{"9","Sheldon","11199230577","sheldor1","notmyspot","4,Caltech Estate","Ghatkopar"},

{"10","Jasmine","56484848454","jasmine3","iamaflower","8,Jumping Street","Dadar"}};

/\*Adding books to the database

Now here we have a matrix with rows=15 and coloumns=5

Here number of rows is the number of books and number of coloumns is the criterias for database, they are as follows:

0->Customer Id of the customer who is selling the book

1->Book Code

2->Title of the book

3->Genre

4->Price

\*/

string books[20][5]={

{"5 ","245","The Murder On The Orient Express ","Crime","299"},

{"6 ","160","Much Ado About Nothing ","Drama","159"},

{"2 ","999","Harry Potter And Goblet Of Fire ","Fantasy","345"},

{"6 ","307","The Mysterious Affair At Styles ","Crime","199"},

{"8 ","419","Around The World In Eighty Days ","Action","139"},

{"7 ","118","The Haunting Of The HillHouse ","Horror","449"},

{"1 ","652","How To Train Your Dragon ","Fantasy","320"},

{"3 ","212","The Adventures Of Tom Sawyer ","Drama","599"},

{"4 ","456","Strange Case Of Missing Cards ","Horror","259"},

{"1 ","111","The Cards On The Murder Table ","Crime","100"},

{"1 ","529","Diary Of Madam Annie Scott ","Fantasy","799"},

{"9 ","379","Buddenbrooks:The Declined Call ","Drama","320"},

{"2 ","291","Legend Of The Sleepyhollow ","Horror","699"},

{"8 ","535","The Girl With The Dragon Tattoo ","Action","199"},

{"9 ","700","Harry Potter And Cursed Child ","Crime","799"},

{"5 ","167","Harry Potter And Philosopher's Stone ","Fantasy","699"},

{"7 ","100","Harry Potter And Half Blood Prince ","Horror","399"},

{"9 ","200","Harry Potter And Story Continues ","Fantasy","999"},

{"4 ","430","How Does Netlix Make So Much Money ","Drama ","469"},

{"6 ","969","The Sun Outshines All Living Creatures","Action","799"}};

/\*Creating databse of cities which are used in the code

We have 10 localities in our database: Kandivali,Malad,Thane,Powai,Vile

Parle,Worli,Mulund,Vikhroli,Ghatkopar,Dadar\*/

string cities[10]={"Kandivali","Malad","Thane","Powai","VileParle","Worli","Mulund","Vikhroli","Ghatkopar","Dadar"};

/\*Storing the adjacency matrix for the graph

No Of Rows=No of Columns= No of cities

This matrix stores the distances between the places

It a weighted graph\*/

int graphDatabase[10][10]={

{0,8,2147483647,3,4,2147483647,2147483647,2,2147483647,10},

{8,0,7,2147483647,6,2147483647,1,3,2147483647,7},

{2147483647,3,0,8,2147483647,8,2,2147483647,8,2147483647},

{3,2147483647,8,0,2,2,4,2147483647,2147483647,5},

{2,2147483647,2147483647,10,0,2147483647,9,2147483647,2,7},

{10,2147483647,5,2,2147483647,0,2147483647,7,3,8},

{2147483647,1,2147483647,4,2147483647,5,0,9,1,2147483647},

{9,3,4,2147483647,2147483647,4,9,0,2147483647,6},

{2,1,2,2147483647,2147483647,10,3,5,0,2147483647},

{10,2147483647,6,2147483647,7,4,2147483647,3,6,0}

};

//We are done with creating all databases

//Now we will declare all the global variables

//Note that the variable runLoop is for printing lines in the entire project for GUI

int NoOfPlaces=10; //Stores the number of cities/places

int noOfCustomers=10; //Stores the number of customers in the database

int noOfBooks=20; //Stores the number of books in the databse

struct cart \*head=NULL; //head pointer for the linked list //ADMIN lets try to add this in the linked list cart

string name,address,locality,userid,number; //To store the details of the buyer

string sellerLocality; //stores the locality of the seller to be later implemented in admin part

/\*We have 2 main classes in our project:Graph and cartMenu

Graph is the class implementing Dijkstra's Algorithm

cartMenu is the class which is the main implementation of the linked list and has all the funtions of the linked list

\*/

//Class Graph

class Graph{

/\*This is the Graph class which is the implementation of the Dijkstra's algorith

It calculates the shortest distance and path between the seller and buyer.

The functions of of this class are:

setGraph()

BFS()

dijkstra() \*/

public:

friend void getDistance(string s, string e); //We are declaring the function getDistance from class Graph to make it as a friend function

//This function is to set a user defined Graph, that is, if the admin wants to have another graph and not the one whichis predefined

void setGraph()

{

int noOfPlaces1; //Stores number of nodes

char choice; //stores choice for BFS

int m,i,j; //Variables used in the 'for' loops

//Inputing the number of nodes

cout<<"Enter the number of places(nodes): ";

cin>>noOfPlaces1;

noOfPlaces1=10;

//Declaring the arrays

int dij[noOfPlaces1][3]={0};

int graphByUser[10][10];

string places[10];

//Inputing the names of the nodes

cout<<"Enter the places: "<<endl;

for(m=0;m<noOfPlaces1;m++)

{

cout<<"PLACE "<<m<<" : ";

cin>>places[m];

}

//Inputing the distances(matrix)

cout<<"For no edge enter 0(it will be converted to infinity)"<<endl;

cout<<"Enter the matrix: "<<endl;

for(i=0;i<noOfPlaces1;i++)

{

for(j=0;j<noOfPlaces1;j++)

{

cout<<"Enter value at row "<<i<<" and coloumn "<<j<<" :";

cin>>places[i][j];

if(graphByUser[i][j]==0&&i!=j)

graphByUser[i][j]=2147483647; //For the places where there is no path, the matrix value will be set as infinity

}

}

dijkstra(graphByUser,places,sellerLocality,locality,noOfPlaces1); //Calling Dijkstra

//Displaying the graph

cout<<"Do you want to display this graph (Enter y/n):";

cin>>choice;

if(choice=='y')

BFS(0,graphByUser,places,noOfPlaces1);

}//function setGraph() ends here

//This function is the breadth first search to print the graph

void BFS(int vertex,int graph[10][10],string places[10],int noOfPlaces1)

{

int i; //'for' Loop

cout<<endl;

int visited[noOfPlaces1]={0}; //Declaring visited array

//Creating an object of the predefined queue function

queue <int> q;

visited[vertex]==1;

q.push(vertex);

while(!q.empty())

{

vertex=q.front();

q.pop();

cout<<places[vertex]<<" \* ";

for(i=0;i<noOfPlaces1;i++)

{

if(graph[vertex][i]==1)

{

if(visited[i]==0)

{

visited[i]=1;

q.push(i);

}

}

}

}

}//BFS() ends here

//Function to calculate the shortest distance and path

void dijkstra(int graph[10][10],string places[10],string sourceParameter,string endParameter,int noOfPlaces)

{

//declaration of variables

int i,j,k,c,r;//Variables used in the 'for' loops

int vertex,n,small,source,end,counter;

int dij[noOfPlaces][3];//defining the dijkstra's matrix

path \*head=NULL;//traversing the path linked list

//Finding the index of the source and destination

for(c=0;c<noOfPlaces;c++)

{

if(places[c]==sourceParameter)

source=c;

if(places[c]==endParameter)

end=c;

}

//Setting the initial value of the dijkstra's matrix,i.e, visited=0,distance=infinity,parent=0

for(i=0;i<noOfPlaces;i++)

{

dij[i][1]=2147483647;

dij[i][0]=0;

dij[i][2]=0;

}

//setting the distance of the source node as 0

dij[source][1]=0;

for(i=0;i<noOfPlaces;i++)

{

//Finding the smallest distance

small=100,counter;

for(r=0;r<noOfPlaces;r++)

{

if(dij[r][0]==0) //does not consider the nodes which are already visited

{

if(dij[r][1]<=small)

{

small=dij[r][1];

counter=r;

}

}

}

//Setting the node with the smallest distance as the vertex

vertex=counter;

//Making the visited of the vertex as 1

dij[vertex][0]=1;

//The following code snippet performs that part of the algorithm which compares the distances and replaces accordingly

for(j=0;j<noOfPlaces;j++)

{

if(dij[j][0]==0) //Does not consider the nodes which are already visited

{

//Compares the values: goes inside the 'if' only if new value is smaller than the original value and the path exists

if(graph[vertex][j]+dij[j][1]<dij[vertex][1]&&graph[vertex][j]!=2147483647&&j!=vertex)

{

dij[j][1]=graph[vertex][j]+dij[vertex][1]; //sets the new value

dij[j][2]=vertex; //sets the parent

}

}

}

} //the algorithm for dijkstra's ends here

//The following part is to print the shortest path and distance

//Prints the value at the 'distance' coloumn of the matrix

cout<<"Distance : "<<dij[end][1]<<" kms"<<endl;

//To print the path, we use a single linked list

cout<<"Shortest Path : ";

k=end;

//The following code stores the path in the linked list

while(k!=source)

{

path \*newNode=new path;

newNode->nextVertex=places[k];

newNode->next=head;

head=newNode;

k=dij[k][2];

}//insert beg

/\*As we need to include the source in our path, we can use the following code snippet instead of individually printing the source

path \*newNode=new path;

newNode->data=nodes[source];

newNode->next=head;

head=newNode;

\*/

//Printng the source

cout<<places[source]<<"-->";

//Printing the path(i.e. the linked list)

path \*temp=head;

while(temp!=NULL)

{

cout<<temp->nextVertex;

cout<<"-->";

temp=temp->next;

}

cout<<"\*";

}//dijkstra() ends here

};//class Graph ends here

//Class cartMenu

class cartMenu

{

/\*The functions of the class cartMenu are:

countNodes()

add()

deleteEntire()

deleteItem()

deleteOnBookid()

deletDuplicate()

display()

totalAmount()

sortAscending()

sortDescending

connectCustomer()

getDistance()

menu()\*/

public:

friend void showBooksMenu();//We are declaring the function showBooksMenu() as a friend of this class so this class is accessible there

int countNodes()

{

//This function will traverse till the end of the linked list and will count the total number of nodes present

int count=0; //count variable counts the number of nodes (counter variable)

cart \*temp=head;

while(temp!=NULL)

{

count++;

temp=temp->next;

}

return count;

}//Function countNodes ends here

//Function to add an item to the cart

void add(string cCode,string bId,string bTitle,string bGenre,int amt)

{ //This function adds a new node to the list like adding an element to the cart

//The parameters are added to the struct upon creating a new node

cart \*newNode= new cart; //Creating a new node

newNode->price=amt;

newNode->customerId=cCode;

newNode->bookId=bId;

newNode->genre=bGenre;

newNode->title=bTitle;

newNode->next=head;

head=newNode;

}//Function add ends here

//Function to delete the entire cart

void deleteEntire()

{ cart \*temp;

if(head==NULL)

cout<<"Your cart is already empty!"<<endl;

else{

while(head!=NULL)

{ temp=head;

head=head->next;

delete temp;

}

cout<<"Cart Deleted successfully!"<<endl;

}

}//deleteEntire ends here

//Function to remove an item from the cart based on the serial number

void deleteItem(int position)

{ /\*This function deletes the item/node from the cart on the serial number

The user will enter the serial number of the item which will act as the position of the node\*/

int check=1; //counter to find the position of the node

cart \*current=head; //pointer at the current node

cart \*pre=head;//pointer to track the previous node of the current

if(head==NULL)

cout<<endl<<"The cart is empty! You can not delete any items.";

else if(position==1)

{for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}cout<<endl;

cout<<"Item Deleted Successfully!!"<<endl;

cout<<"You just deleted: "<<endl;

cout<<"Product Name: "<<head->title<<endl;

cout<<"Product Code: "<<head->bookId<<endl;

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}

head=head->next;

}

else{

if(countNodes()>=position) //if to check if the position is within the limits

{

check=1;

while(check!=position) //Finds the node to be deleted from the position

{

pre=current;

current=current->next;

check++;

}

//cout<<current->bookId<<endl;

//Displaying the details of the item before deleting it

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}

cout<<"Item Deleted Successfully!!"<<endl;

cout<<"You just deleted: "<<endl;

cout<<"Product Name: "<<current->title<<endl;

cout<<"Product Code: "<<current->bookId<<endl;

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}

//Deleting the node from the memory

pre->next=current->next;

current=NULL;

delete current;

//Linking the nodes again

}

else if(countNodes()<position)

cout<<"The Number does not exist in the cart."<<endl;

}

}//Function deleteItem ends here

//Function to delete the item based on the book idea

void deleteOnBookid(string data)

{

cart \*current=head;

int counter=0;//counter to check if the bookId is actually present in the cart

cart \*pre=head; //tracks the previous node

if(head==NULL)

cout<<"Empty Cart Please Try Again!"<<endl;

else if(head->bookId==data)

{

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}cout<<endl;

cout<<"Item Deleted Successfully!!"<<endl;

cout<<"You just deleted: "<<endl;

cout<<"Product Name: "<<head->title<<endl;

cout<<"Product Code: "<<head->bookId<<endl;

head=head->next;

for(int runLoop=0;runLoop<=100;runLoop++){ cout<<"-";}

}

else{

while(current!=NULL)

{

if(current->bookId==data)

{

counter++; //counter increments if the bookId is in the cart

break;

}

else{

//Traversing the list

pre=current;

current=current->next;

}

}

if(counter==1)

{//Deleting the item

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}cout<<endl;

cout<<"Item Deleted Successfully!!"<<endl;

cout<<"You just deleted: "<<endl;

cout<<"Product Name: "<<current->title<<endl;

cout<<"Product Code: "<<current->bookId<<endl;

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"-";}

pre->next=current->next;

current=NULL;

delete current;

}

else if(counter==0)

cout<<"The BookId you are trying to delete is not in the cart!"<<endl;

}

}//deleteOnBookId ends here

//Function to delete duplicates in the list

void deleteDuplicate()

{

cart \*temp=head;cart \*temp2;

cart \*temp1;

if(head==NULL)

cout<<"Empty Cart! No items to check for duplicates!"<<endl;

else {

while(temp!=NULL)//traversing through the entire list to check duplicates for each element

{

temp1=temp->next;

while(temp1!=NULL)//traversing through the entire list to check duplicates of temp

{

if(temp->bookId==temp1->bookId)//checking for duplicates

{ //deleting if duplicate found

temp2=temp1;

temp->next=temp1->next;

temp2=NULL;

delete temp2;

}

temp1=temp1->next;

}

temp=temp->next;

}

}

}//deletDuplicates() ends here

//This function is to display the items of the cart

void display()

{

int serialNo=0;//Counter variable to print serial number

cart \*temp=head;

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"~";}

cout<<endl<<"\t\t HI THERE! WELCOME TO YOUR CART!"<<endl;

for(int runLoop=0;runLoop<=100;runLoop++)

{ cout<<"~";}

cout<<endl<<"Here are the items of your cart: "<<endl;

if(head==NULL)

cout<<"OOPS! Your cart is empty!"<<endl;

else{

cout<<"NO\tBOOKCODE\tBOOK NAME\t\t\tGENRE\tPRICE\t"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}

cout<<endl;

while(temp!=NULL)

{

serialNo++;

cout<<serialNo<<")\t"<<temp->bookId<<" \t "<<temp->title<<" \t"<<temp->genre<<" \t"<<temp->price<<endl;

temp=temp->next;

}

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

cout<<"Details of your cart: "<<endl;

cout<<"Total Items : "<<countNodes()<<endl;

cout<<"Total Amount : Rs."<<totalAmount()<<endl;

for(int i=0;i<100;i++){cout<<"-";}

cout<<endl;

}

}//Function Display ends here

//Function to calculate the total amount of the cart

int totalAmount()

{ int total=0;

cart \*temp=head;

while(temp!=NULL)//traversing the entire list

{

total+=temp->price;//adding the amount

temp=temp->next;

}

return total;

}//Function total amount ends here

//Function to sort the cart in ascending order

void sortAscending()

{

cart\* curr = head;

int count = 0;

for(int i = countNodes() ; i > 1 ; i-- )

{

cart \*temp, \*swap1;

swap1 = head;

for(int j = 0 ; j < countNodes()-1 ; j++ )

{

if(swap1->price > swap1->next->price)

{

cart \*swap2 = swap1->next;

swap1->next = swap2->next;

swap2->next = swap1;

if(swap1 == head)

{

head = swap2;

swap1 = swap2;

}

else

{

swap1 = swap2;

temp->next = swap2;

}

}

temp = swap1;

swap1 = swap1->next;

}

}

} //Sort Function for ascending ends here

void sortDescending()

{

cart\* curr = head;

int count = 0;

for(int i = countNodes() ; i > 1 ; i-- )

{

cart \*temp, \*swap1;

swap1 = head;

for(int j = 0 ; j < countNodes()-1 ; j++ )

{

if(swap1->price < swap1->next->price)

{

cart \*swap2 = swap1->next;

swap1->next = swap2->next;

swap2->next = swap1;

if(swap1 == head)

{

head = swap2;

swap1 = swap2;

}

else

{

swap1 = swap2;

temp->next = swap2;

}

}

temp = swap1;

swap1 = swap1->next;

}

}

}//sortFunction for Descending ends here

//This function connects the book to its seller and displays the details of the customer selling the book

void connectCustomer()

{

int counter=0;

int i,j;//'for' loops

int seller; string buyer; //storing buyer and seller

cout<<endl<<endl<<endl;

for(int runLoop=0;runLoop<=100;runLoop++){cout<<"~";}cout<<endl;

for(int runLoop=0;runLoop<=100;runLoop++){cout<<"~";}cout<<endl;

cout<<"\tFINAL BILL"<<endl;

for(int runLoop=0;runLoop<=100;runLoop++){cout<<"~";}cout<<endl;

cout<<"Welcome to the check out counter!"<<endl;

cout<<"Now we will connect you to your user from whom you want to buy the book!"<<endl;

for(int runLoop=0;runLoop<=100;runLoop++){cout<<"-";}cout<<endl;

//Prints details of the cutomer

cout<<"User Name: "<<name<<endl;

cout<<"User Number: "<<number<<endl;

cout<<"User Address: "<<address<<endl;

cout<<"User Locality: "<<locality<<endl;

cout<<endl;

for(int runLoop=0;runLoop<=100;runLoop++){cout<<"-";}cout<<endl;

//Prints details of the books

cout<<"\tBOOK AND SELLER DETAILS"<<endl<<endl;

cart \*temp2=head;

while(temp2!=NULL)//Traversing the linked list for the cart

{

counter++; seller=0;

cout<<endl<<"BOOK "<<counter<<": "<<endl;

for(int runLoop=0;runLoop<=30;runLoop++){cout<<"-";}cout<<endl;

cout<<"Title : "<<temp2->title<<endl;

cout<<"Book Id : "<<temp2->bookId<<endl;

cout<<"Genre : "<<temp2->genre<<endl;

cout<<"Price : "<<temp2->price<<endl;

buyer=locality;

//The Following code snippet is to convert the string value of the customer to integer

for(int j=0;j<1;j++)

{

char c=temp2->customerId[j]; //customerId the member of the structure cart

int n=c-48;

seller=seller\*10+n;

}

seller=seller-1;

//Printing the details of the seller

cout<<"Name of seller : "<<customers[seller][1]<<endl;

cout<<"Phone Number : "<<customers[seller][2]<<endl;

cout<<"Address : "<<customers[seller][5]<<endl;

cout<<"Loacality : "<<customers[seller][6]<<endl;

sellerLocality=customers[seller][6];

//calling getDistance to display the distance and the shortest path between the seller and the buyer

getDistance(buyer,customers[seller][6]);

temp2=temp2->next;

cout<<endl;

}

counter=0;

cout<<endl<<"~~~~~~~~~~~"<<endl;

cout<<"TOTAL AMOUNT YOU WILL HAVE TO PAY: "<<totalAmount()<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";} cout<<endl;

}//connectCustomerEnds here

//function to call dijkstra and get the distance

void getDistance(string buy,string sell)

{

Graph graphObject; //Creating an object of the graph class

graphObject.dijkstra(graphDatabase,cities,buy,sell,NoOfPlaces);

}//function getDistance() ends here

//Menu of the cart

void menu()

{

//This functions calls all the other functions in the class according to the menu

int choice,value;

string bId;

cartMenu c; //Creating object for the class cart

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

cout<<"Welcome to your CART! "<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

//Printing the menu

cout<<"1)Delete item using serial number"<<endl;

cout<<"2)Sort items from lowest to highest(with respect to price)"<<endl;

cout<<"3)Sort items from highest to lowest(with respect to price)"<<endl;

cout<<"4)Display cart"<<endl;

cout<<"5)Display Total Amount and Number of items in the cart"<<endl;

cout<<"6)Delete book by book Id"<<endl;

cout<<"7)Delete entire cart"<<endl;

cout<<"8)Accidentally added double items?Delete Duplicates from the cart"<<endl;

cout<<"9)Proceed to check out counter"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}

for(;;)

{

cout<<endl<<"ENTER YOUR CHOICE: ";

cin>>choice;

cout<<endl;

if(choice==9)

{

c.connectCustomer();

cout<<endl;

break;

}

else

{

switch(choice)

{

case 1:

cout<<"Enter the Sr. No. of the item which you want to delete: ";

cin>>value;

c.deleteItem(value);

cout<<endl;

break;

case 2:

c.sortAscending();

cout<<endl;

break;

case 3:

c.sortDescending();

cout<<endl;

break;

case 4:

c.display();

cout<<endl;

break;

case 5:

if(head==NULL)

cout<<"Looks like your cart is Empty!"<<endl;

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

cout<<"Details of your cart: "<<endl;

cout<<"Total Items : "<<c.countNodes()<<endl;

cout<<"Total Amount : Rs."<<c.totalAmount()<<endl;

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

break;

case 6:

cout<<"Enter the book ID of the item you wish to delete: ";

cin>>bId;

c.deleteDuplicate();

deleteOnBookid(bId);

break;

case 7:

c.deleteEntire();

cout<<endl;

break;

case 8:

c.deleteDuplicate();

break;

default:

cout<<"OOPS! Wrong Option! Please try again!";

break;

}

}

}

}//menu() function ends here

}; //Class cartMenu ends here

//Now we have all the normal functions

/\*The Normal Functions are as follows:

displayCustomers()

displayBooks()

displayGraph()

loginAccount()

showBooksmenu()

DFS()

admin()

main()

\*/

//This function is to display the customer database

void displayCustomers()

{ //This function is to display the customer database

int i,j; //Variables for the 'for' loop

cout<<"Your database is: "<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

cout<<"ID\tNAME\tNUMBER\t\tUSERNAME\tPASSWORD\tADDRESS\t\t\tLOCALTIY\t"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

for(i=0;i<10;i++)

{

for(j=0;j<7;j++)

{

cout<<customers[i][j]<<"\t";

}

cout<<endl;

}

}//Display Customers ends here

//This function is to display the books database

void displayBooks()

{

int i,j; //Variables for the 'for' loop

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

cout<<"NO\tBOOKCODE\tBOOK NAME\t\t\tGENRE\tPRICE\t"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

for(i=0;i<20;i++)

{

for(j=0;j<5;j++)

{

cout<<books[i][j]<<"\t";

}

cout<<endl;

}

}//displayBooks ends here

//This function is to print the adjacency matrix of the graph

void displayGraph()

{

int i,j;

cout<<"The cities in our graph are: "<<endl;

for(i=0;i<NoOfPlaces;i++)

{

cout<<i+1<<". "<<cities[i]<<endl;

}

cout<<endl;

cout<<"For the simplicity of viewing the graph,infinity is displayed as '-1'!"<<endl;

cout<<"The adjacency matrix of graph is as follows: "<<endl;

for(i=0;i<NoOfPlaces;i++)

{

for(j=0;j<NoOfPlaces;j++)

{

if(graphDatabase[i][j]==2147483647)

cout<<"-1 ";

else

cout<<" "<<graphDatabase[i][j]<<" ";

}

cout<<endl;

}

}//displayGraph() ends here

//In this function we check if the user is already a customer, if not then we take his details

void loginAccount()

{

string username,password;

int counter=0;

char loginUser; //Counter to check if the user is registered or not

bool flag;

int choice;

flag=false;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"\t\t\tPICK\_A\_BOOK"<<endl;

cout<<"\tHello dear user,Welcome to \* PICK-A-BOOK \* Book Renting Portal"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"You can rent some really amazing books here from us and enjoy the bliss of reading in a very cost efficient way!"<<endl;

cout<<"We are open now and at your service!!!!"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"Are you a registered user?(enter y or n): ";

cin>>loginUser;

if(loginUser=='y'){

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

do{

cout<<"HELLO USER!Enter your username and password to login to your account"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"Username:";

cin>>username;

//comparing this username with database

for(int i=0;i<10;i++){

if(username==customers[i][3])

{

cout<<"Password:";

cin>>password;

//compare with this username in the database,both should match(username and password to serial no)

if(password==customers[i][4])

{

cout<<endl<<"You were successful in logging in!"<<endl;

flag=true;

name=customers[i][1];

address=customers[i][5];

locality=customers[i][6];

number=customers[i][2];

userid=customers[i][0];

}

}

}

if(flag==false)

{

cout<<"Invalid username or password!Please re-enter!"<<endl;

counter++; //after 5 failed attempts to login,the system will stop asking

}

if(counter==5)

break;

}while(flag==false);

}

else

{

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"Hey There! New user!Welcome to PICK-A-BOOK"<<endl;

for(int runLoop=0;runLoop<120;runLoop++){cout<<"~";}

cout<<endl;

cout<<"Enter the following sign up details to create account"<<endl;

cout<<"Name:";

cin>>name;

cout<<"Number:";

cin>>number;

cout<<"Address:";

cin>>address;

cout<<"Locality(Please Enter from the following ONLY: Kandivali,Malad,Thane,Powai,Vile Parle,Worli,Mulund,Vikhroli,Ghatkopar,Dadar):";

cin>>locality;

}

if(counter==5)

{

cout<<endl;for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

cout<<"You have entered the incorrect username/password 5 times. We can not let you proceed further!"<<endl;

cout<<endl;for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

return;

}

else

showBooksMenu();

//}

} //login account ends here

void showBooksMenu()

{

int i,j; //'for' loop

int counterOfBooks=0,k,amt=0;

string choiceOfGenre,addToCart;

cartMenu cobj; char choice;

cart \*temp2; char c;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

cout<<"\t\tWELCOME USER! BUY A BOOK!"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"~";}cout<<endl;

do{

addToCart="";amt=0;

cout<<endl<<"Which Genre of books do you want buy?We have the following genres: Crime,Horror,Fantasy,Drama,Action."<<endl;

cout<<endl<<"Enter the genre you would like to explore today?(Make sure all the entries match the ones shown above(This is a case sensitivecode)!): ";

cin>>choiceOfGenre;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

//"if" checks if the genre is actually present in the databse

if(choiceOfGenre=="Crime"||choiceOfGenre=="Drama"||choiceOfGenre=="Fantasy"||choiceOfGenre=="Action"||choiceOfGenre=="Horror")

{

cout<<"|Id\t\tTitle\t\t\t\tGenre\tPrice"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

//code snippet to display the books of the entered genre

for(i=0;i<noOfBooks;i++)

{

if(choiceOfGenre==books[i][3])

{

cout<<"|";

for(j=1;j<5;j++)

{

cout<<books[i][j]<<"\t";

}

cout<<"|";

cout<<endl;

}

}

for(int runLoop=0;runLoop<100;runLoop++){cout<<"-";}cout<<endl;

cout<<endl;

cout<<"Please enter the codes of the books you want to buy and enter 'exit' to exit:"<<endl; //input of the code which should be added to cart

for(i=0;addToCart!="exit";i++)

{

cin>>addToCart;

amt=0;

for(k=0;k<noOfBooks;k++)

{

if(books[k][1]==addToCart)

{

//The following code snippet is to convert the amount from string to integer

string s=books[k][4];

//price the member of the structure cart

for(int t=0;t<3;t++)

{

int n=s[t]-48;

amt=amt\*10+n;

}

cobj.add(books[k][0],books[k][1],books[k][2],books[k][3],amt);

break;

}

}

}

}

else{

cout<<"OOPS YOU ENTERED A GENRE WHICH WE DO NOT HAVE!!!! PLEASE TRY AGAIN!"<<endl;

choice='y';

}

cout<<"Do you wish to continue viewing books by genres?(Enter y or n):";

cin>>choice;

}while(choice=='y');

cobj.menu();

}//showBooksMenu() ends here

//function to print the preDefined graph using DFS

void DFS(int vertex)

{

int visited[NoOfPlaces]={0};

int i;//'for' loop

stack<int> s; //creating object of predefined stack

s.push(vertex);

while(!s.empty()) //running till stack is empty

{

vertex=s.top();

s.pop();

if(visited[vertex]==0) //checking for visited

{

visited[vertex]=1;

cout<<cities[vertex]<<" \* ";

for(i=0;i<NoOfPlaces;i++)

{

if(graphDatabase[vertex][i])//checking if path exists

s.push(i);

}

}

}

}//function DFS() ends here

//Function to give admin options which are not available to the user

void admin()

{

Graph objOfGraph;//Creating object of graph class

int adminChoice;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

cout<<endl<<"\t\t\tADMIN SECTION"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

cout<<"As an admin do you want to perform the following the operations?: "<<endl;

cout<<endl<<"Welcome Admin!"<<endl;

cout<<"Please take a look at the menu: "<<endl;

cout<<"1->View Customer Database"<<endl;

cout<<"2->View Book Database"<<endl;

cout<<"3->View the Graph Databse"<<endl;

cout<<"4->Display the Pre-Defined Graph"<<endl;

cout<<"5->Enter a new Graph Database"<<endl;

cout<<"6->Exit"<<endl;

for(;;)

{

cout<<endl<<"Please enter your choice: ";

cin>>adminChoice;

cout<<endl;

if(adminChoice==6)

break;

else{

switch(adminChoice)

{

case 1:

displayCustomers();

break;

case 2:

displayBooks();

break;

case 3:

displayGraph();

break;

case 4:

cout<<endl;

DFS(0);

break;

case 5:

cout<<"We will directly send you to setGraph and from there you can implement Dijkstra's algorithm"<<endl;

cout<<"The function of Dijkstras's Algorithm is independent of the database."<<endl;

cout<<"As per the customer who was logged in and the the details of the seller of the last book,the source and destination are:"<<endl;

cout<<"Source: "<<locality<<endl;

cout<<"Destination: "<<sellerLocality<<endl;

cout<<endl;

objOfGraph.setGraph();

break;

default:

cout<<"You have entered the wrong choice! Please try again later!"<<endl;

}

}

}

}//function admin() ends here

//Main function to call the function

int main()

{

loginAccount();

cout<<"HOPE YOU HAD FUN USING OUR PORTAL!!!"<<endl;

cout<<"THANK YOU FOR BEING OUR CUSTOMERS!!!"<<endl;

for(int runLoop=0;runLoop<100;runLoop++){cout<<"\*";}cout<<endl;

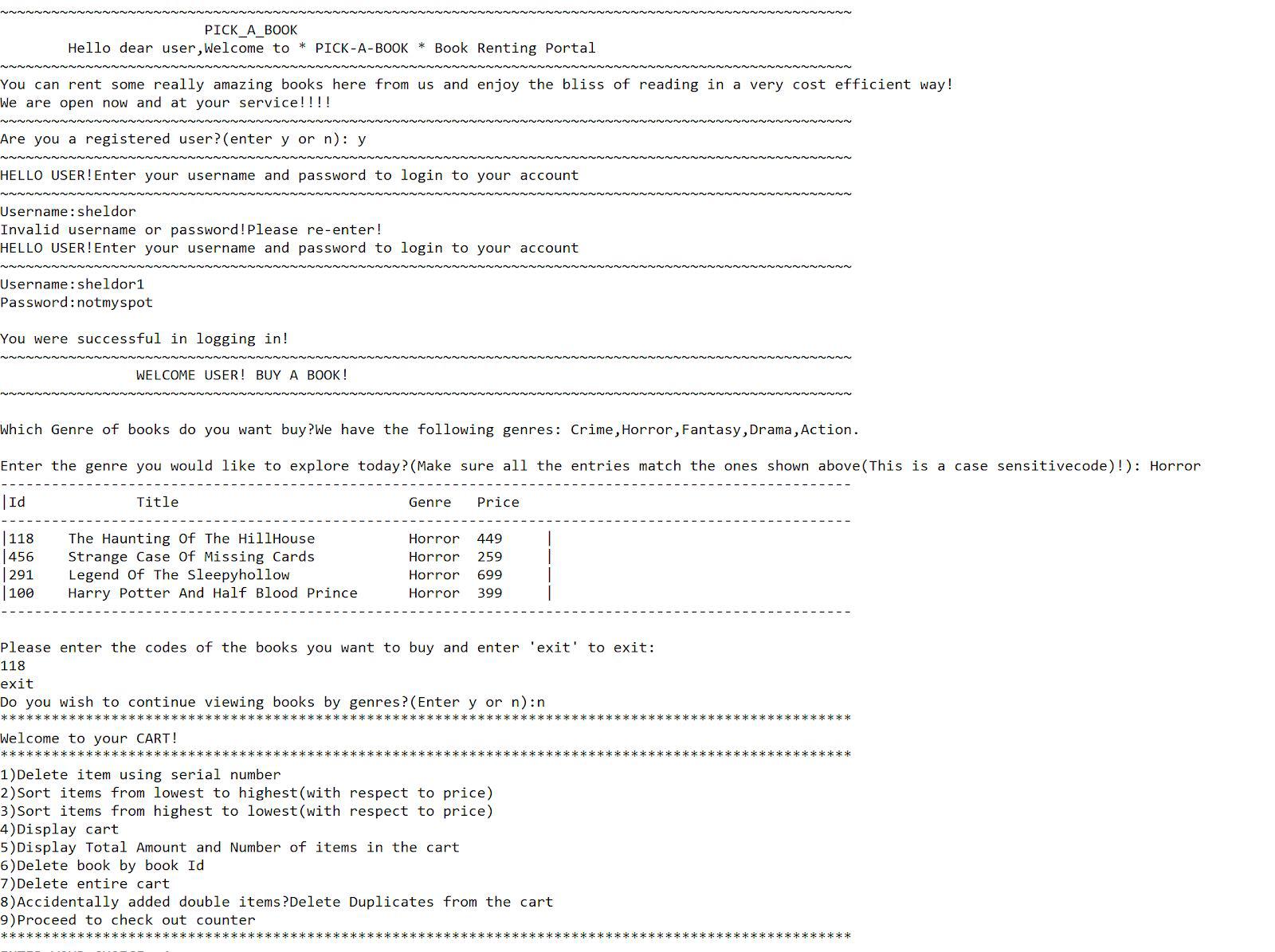
admin();

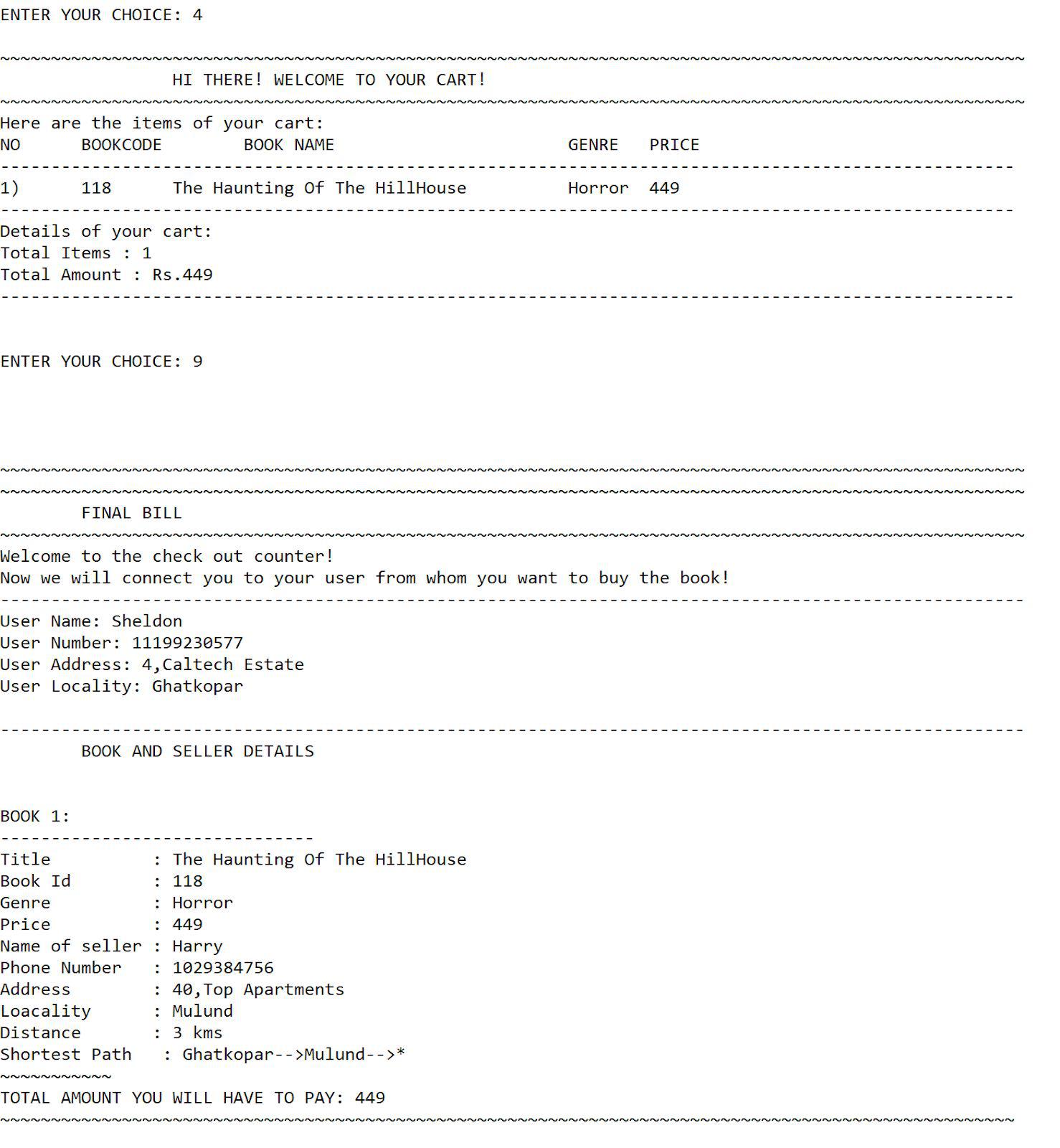
cout<<"END OF CODE";

}

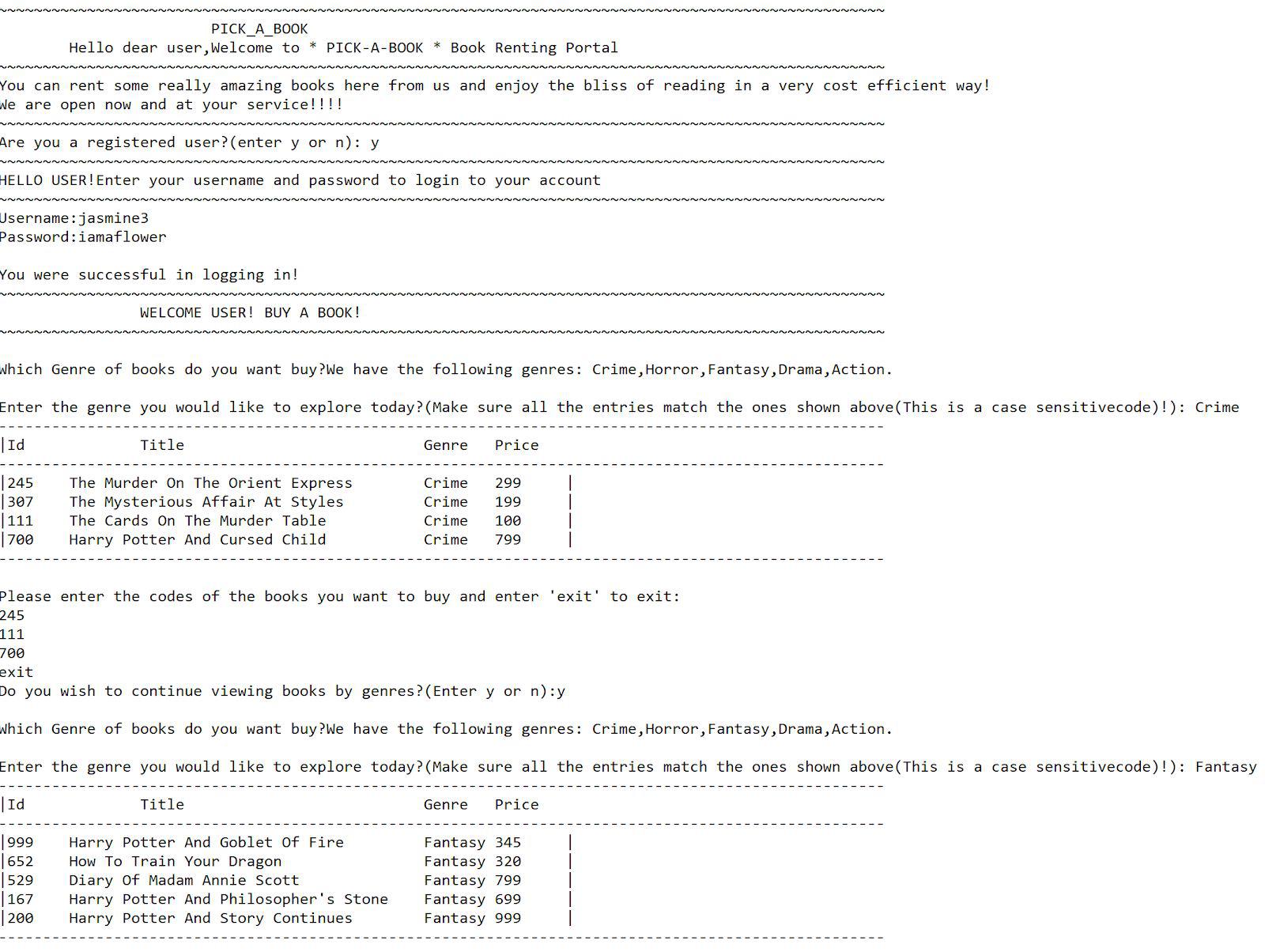
A.7 **Input and Output:**

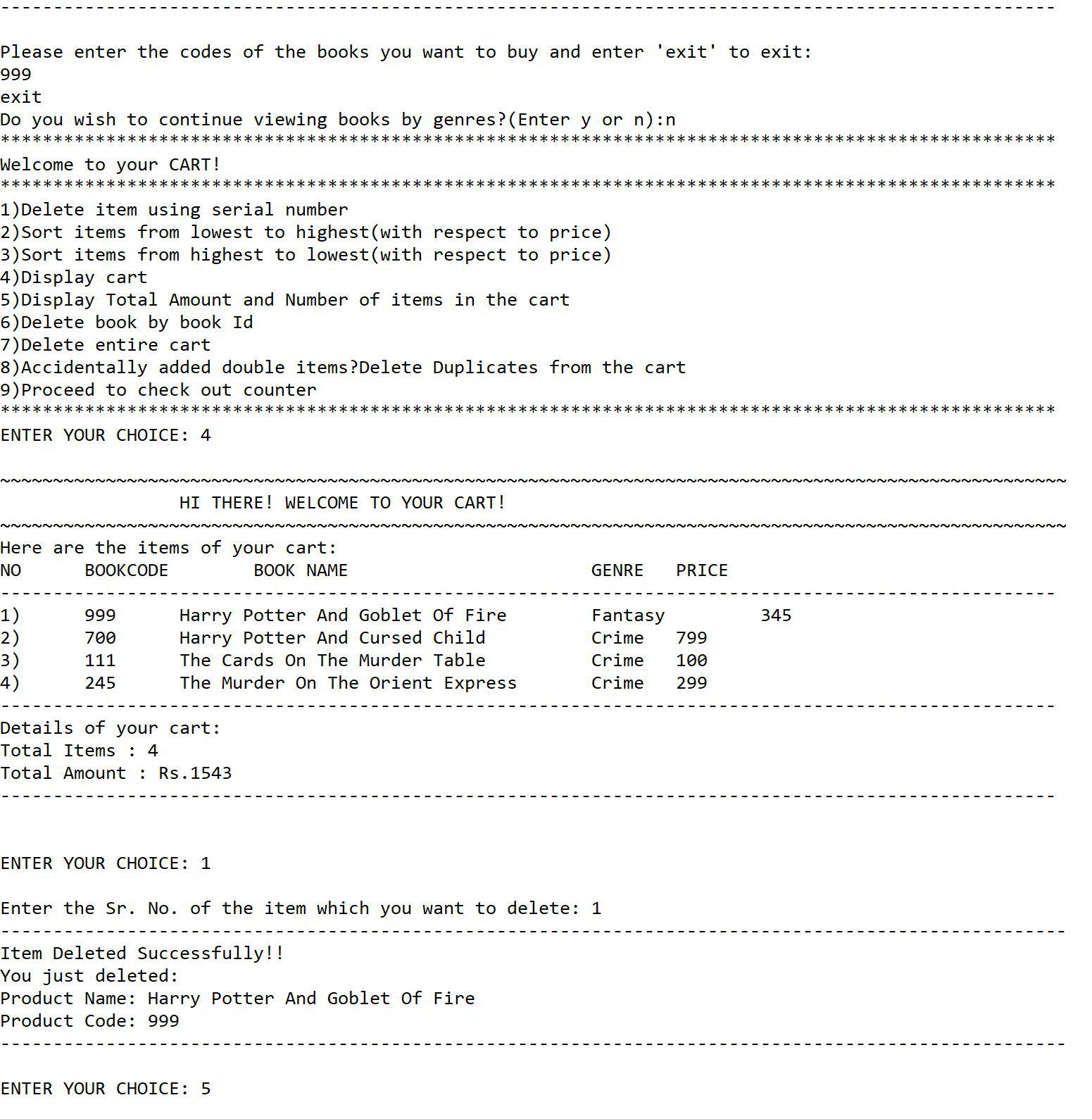
**Output Case 1:**

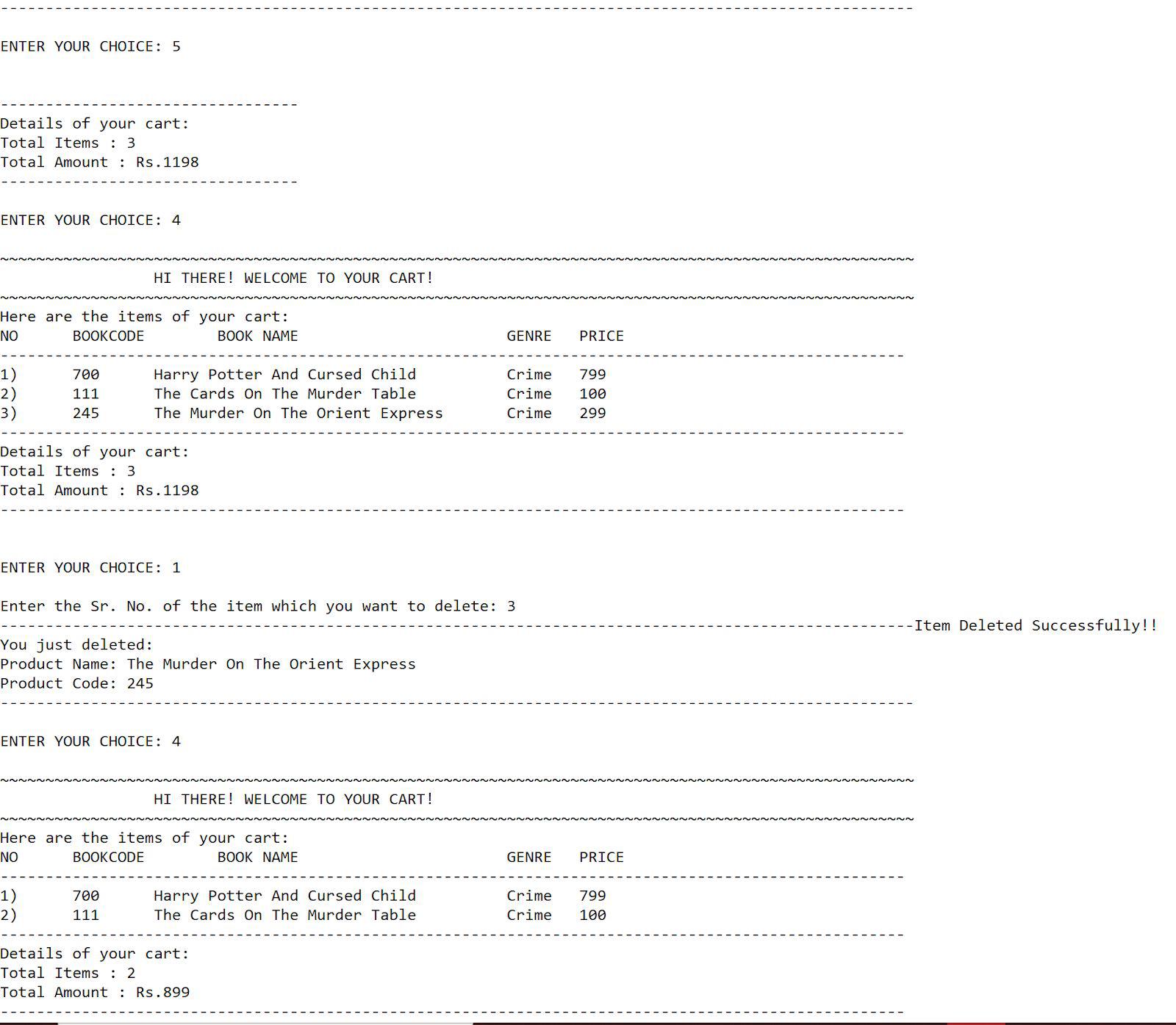


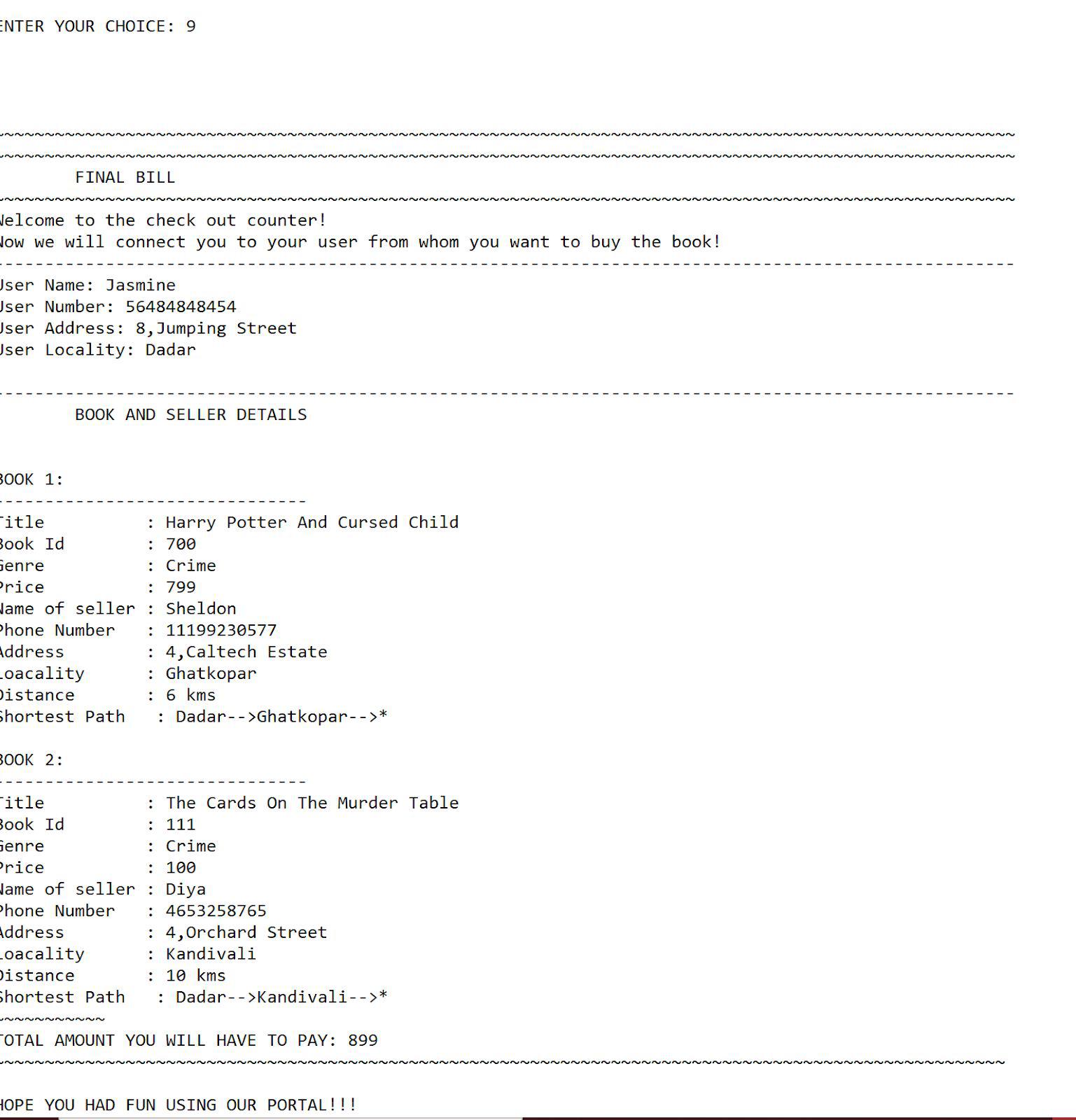


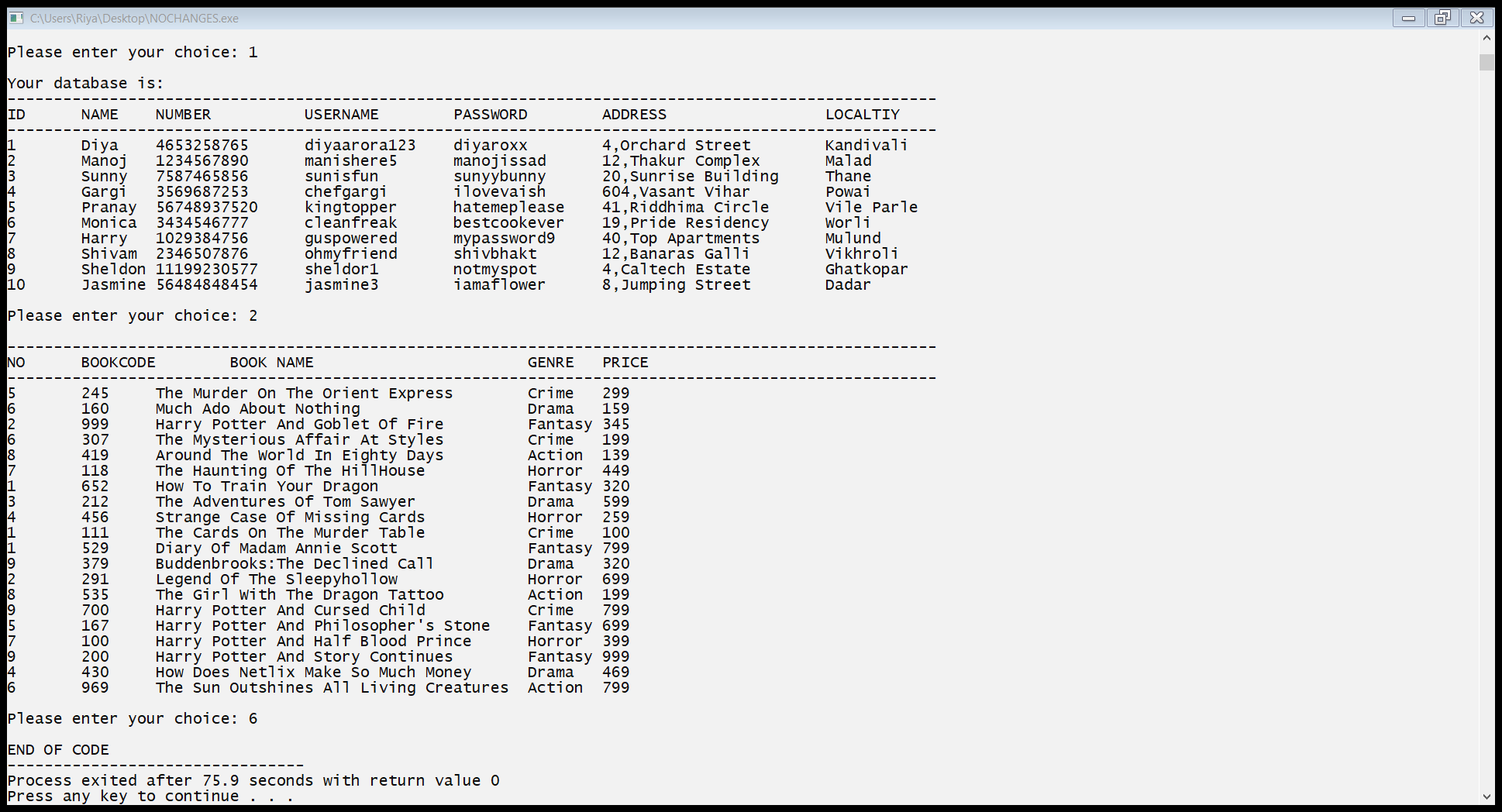
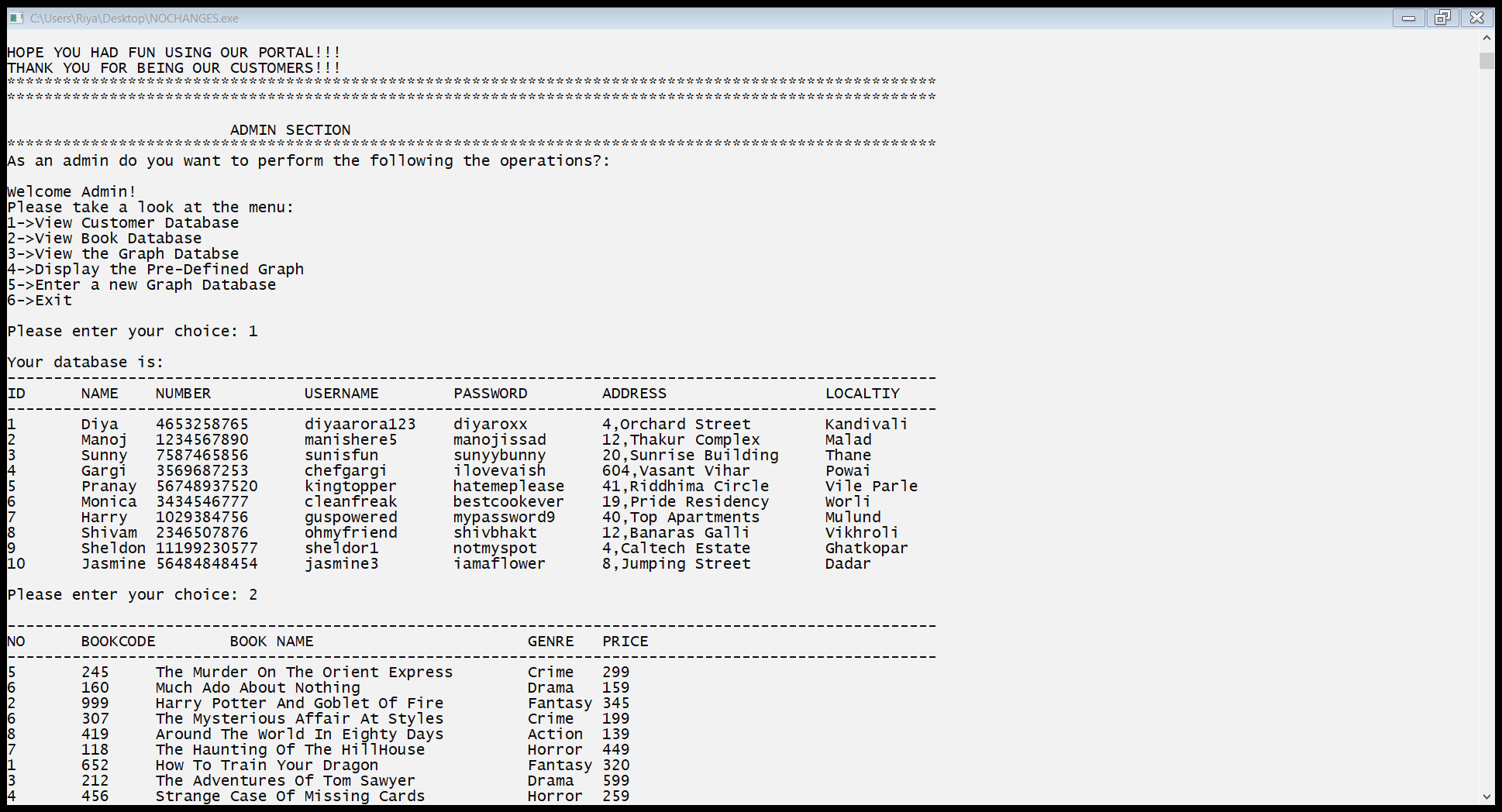
**OUTPUT CASE 2:**



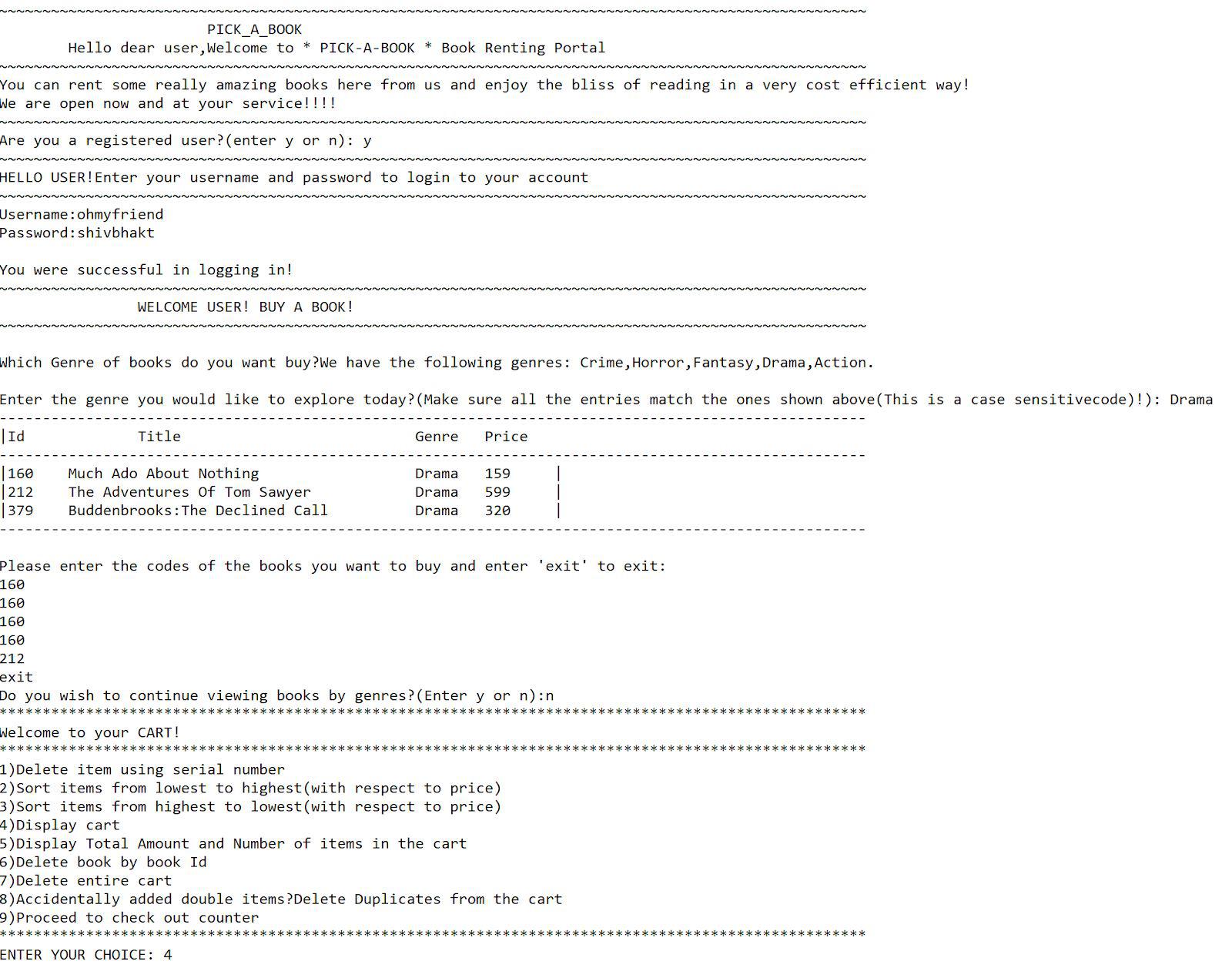


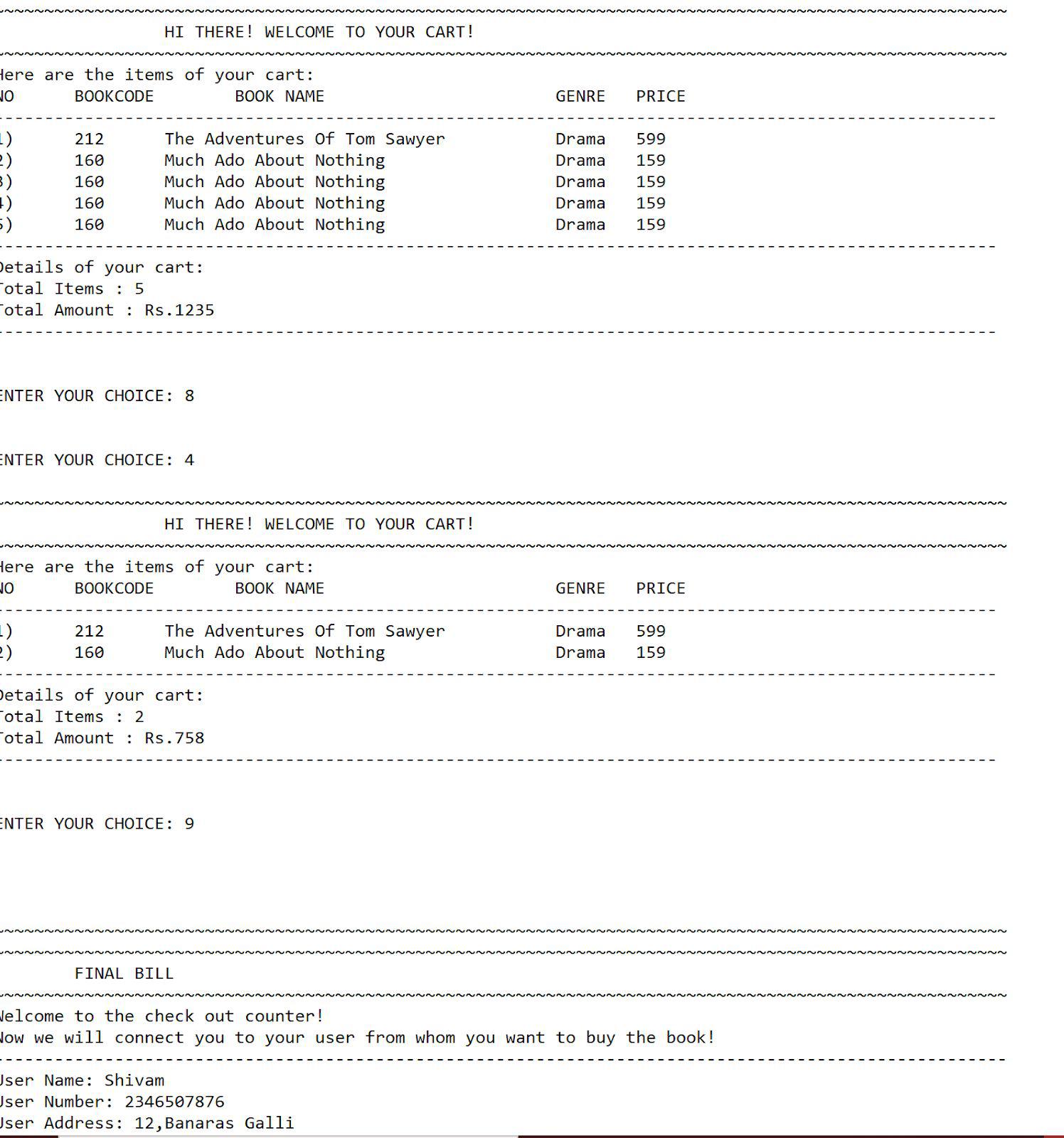


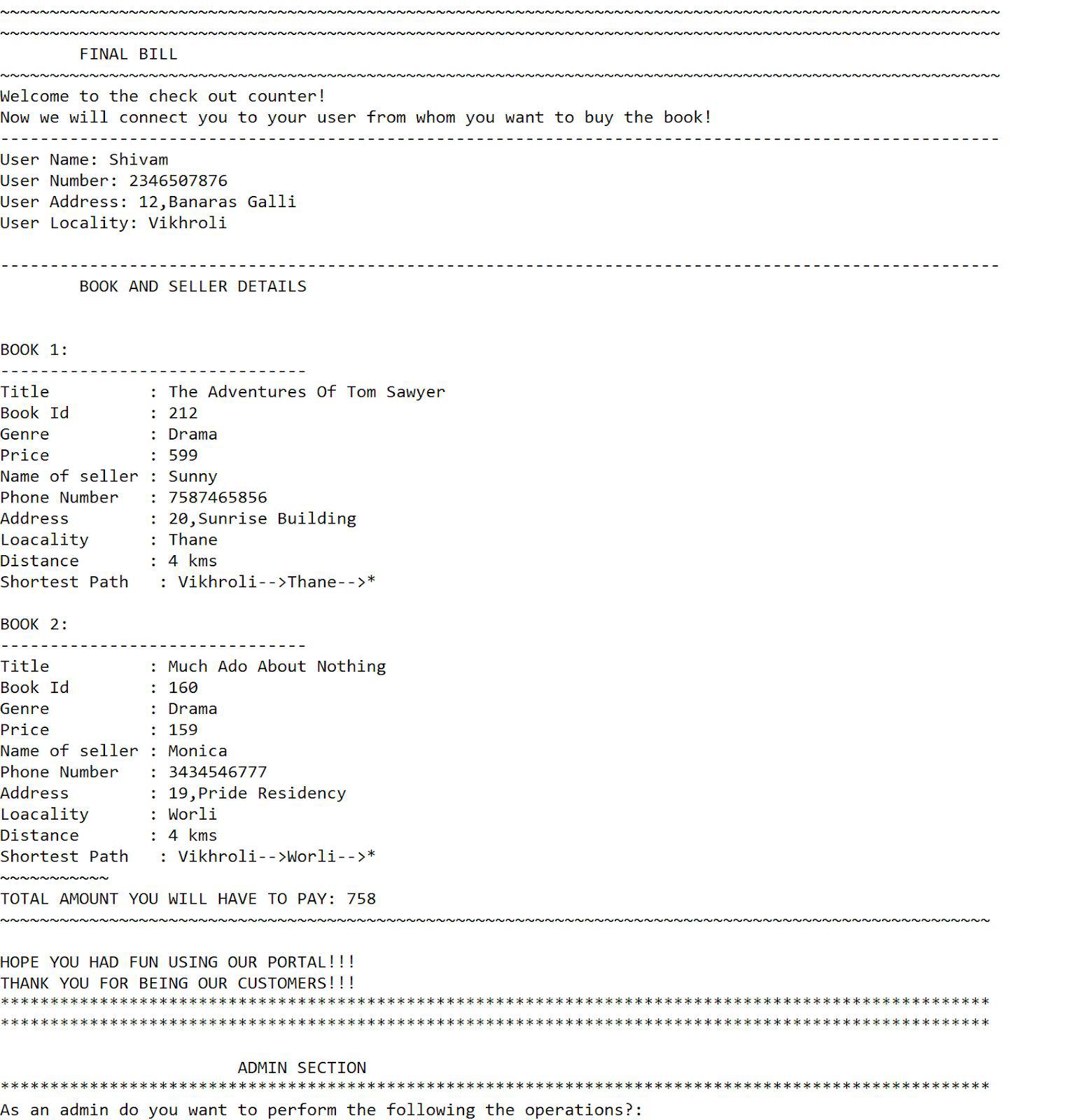


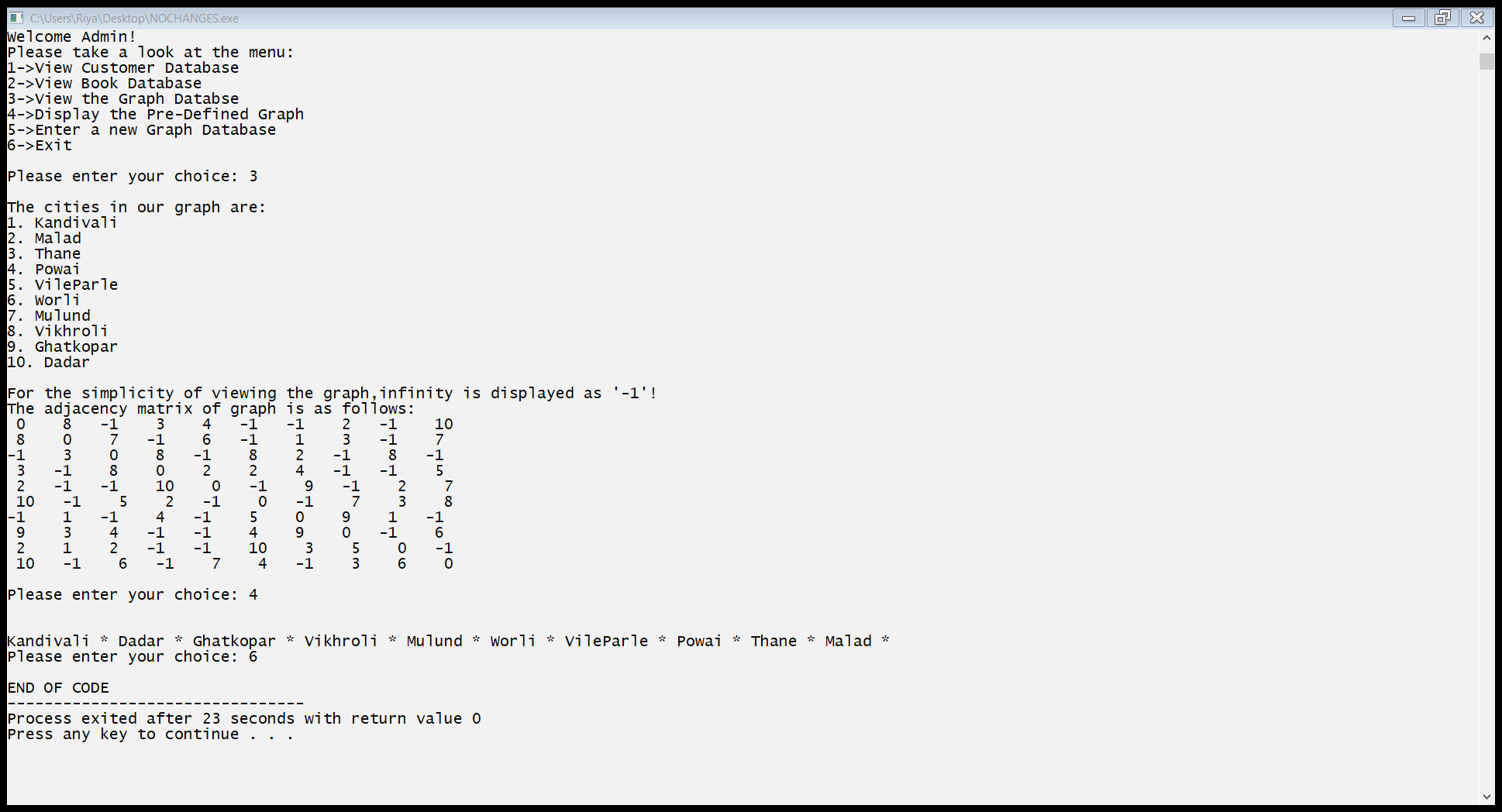
****

**OUTPUT CASE 3:**

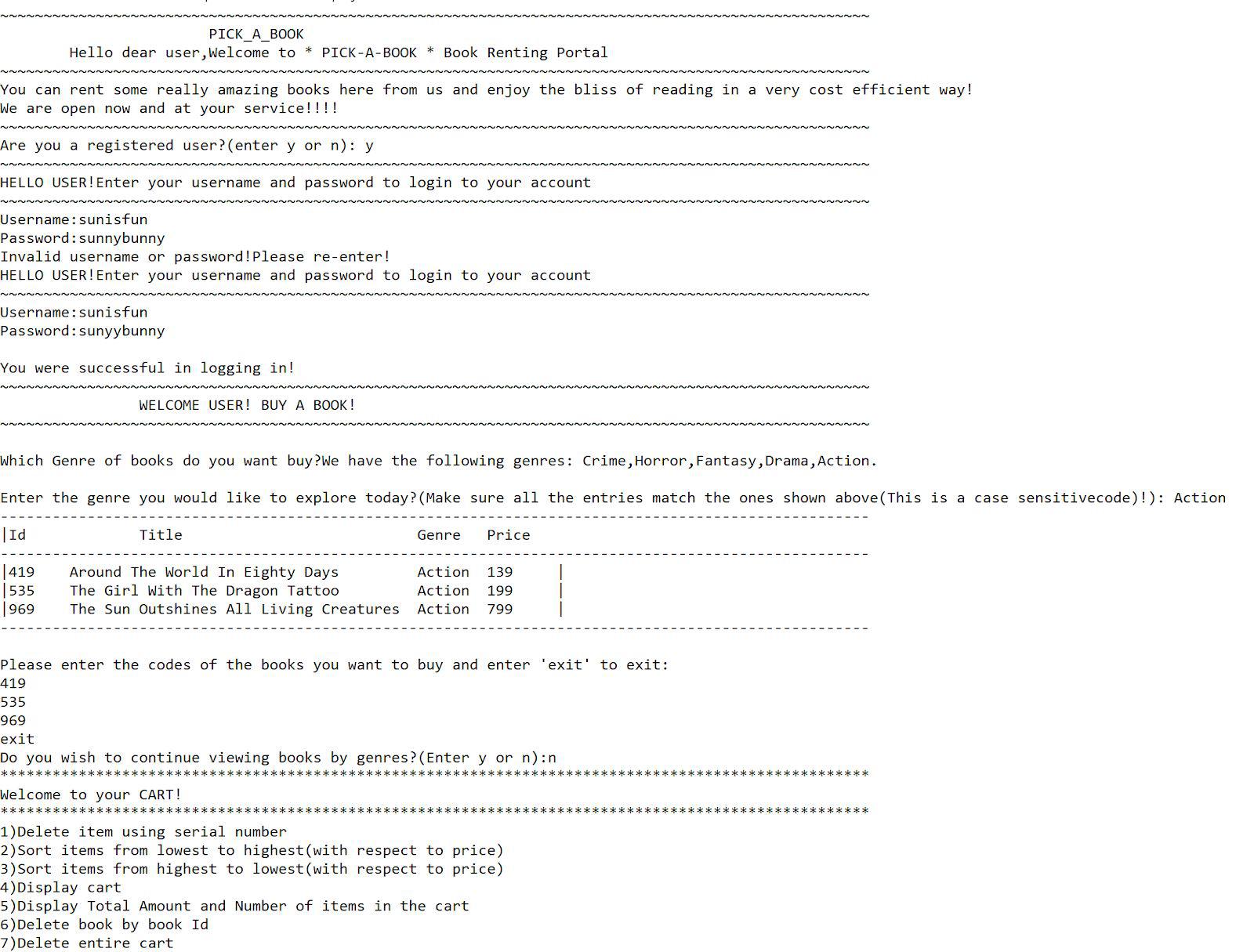


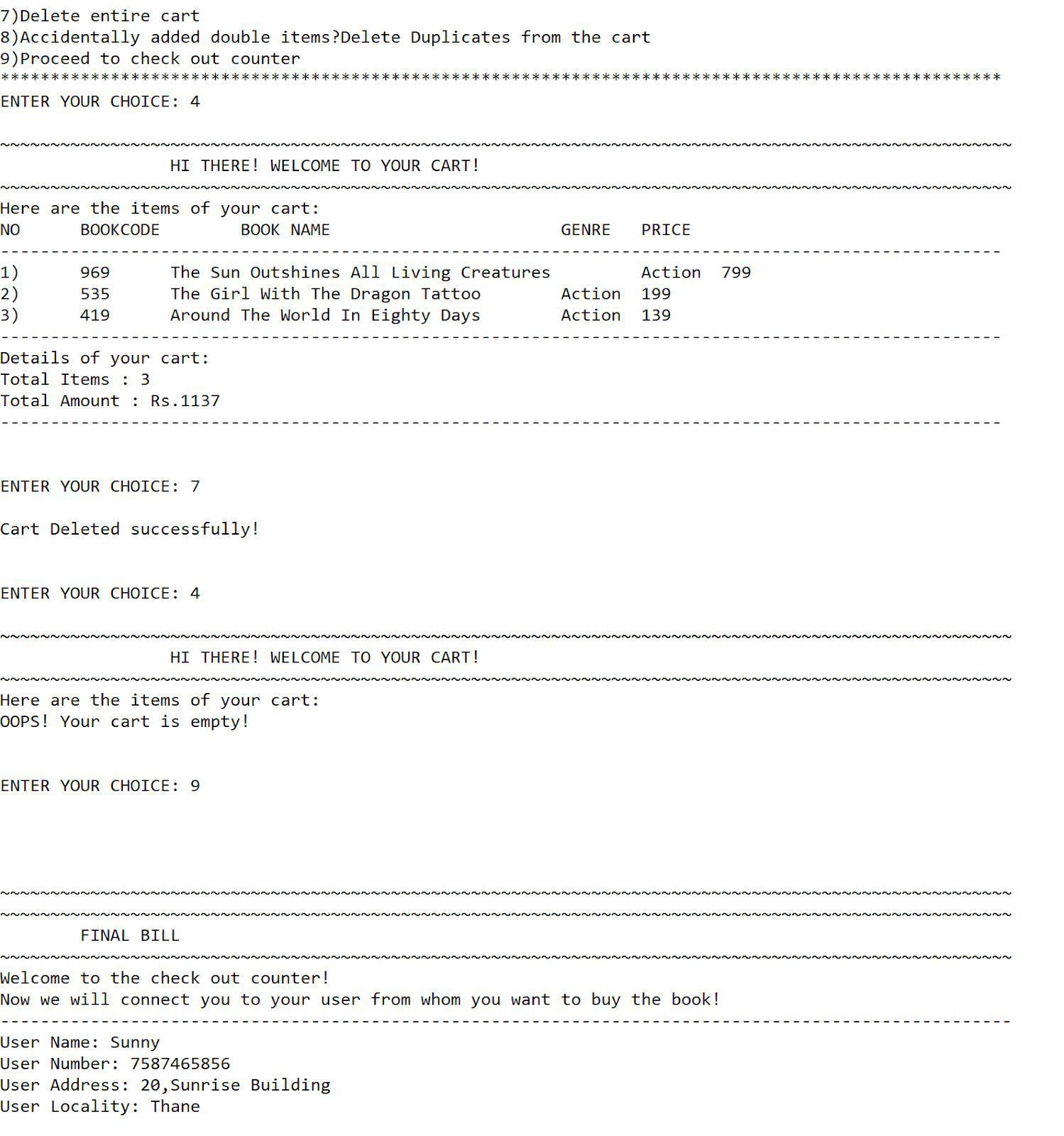


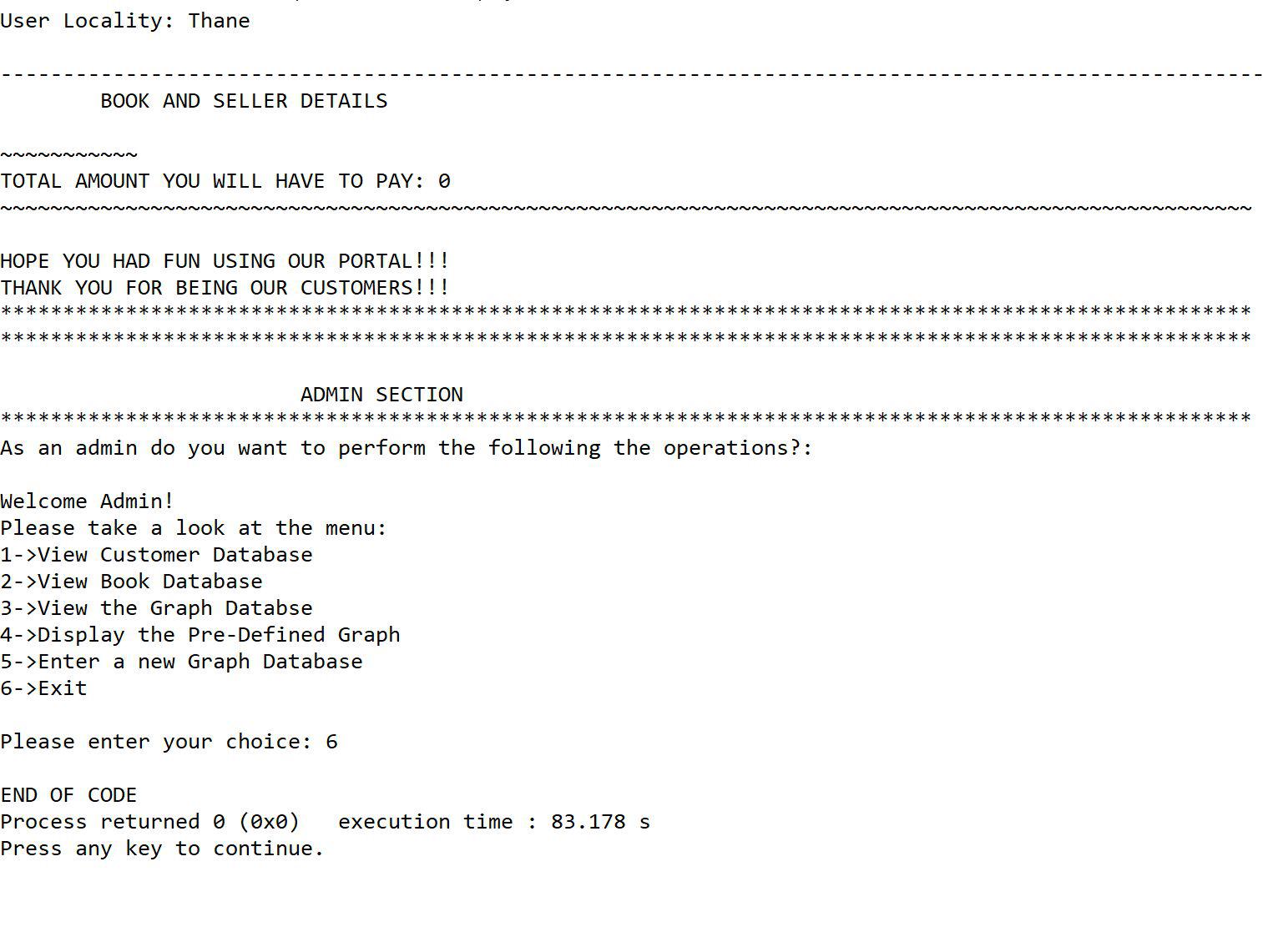


****

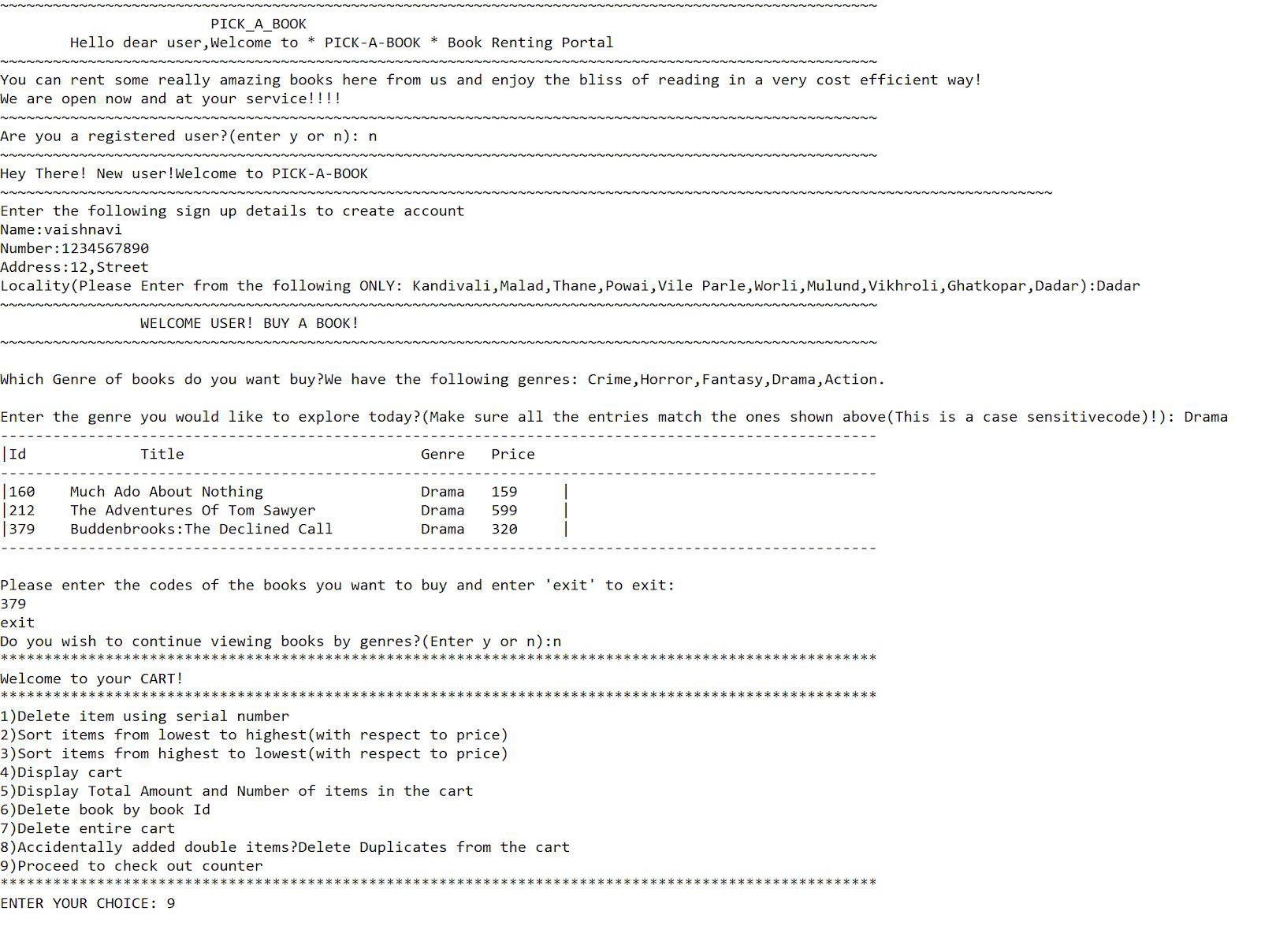
**OUTPUT CASE 4:**





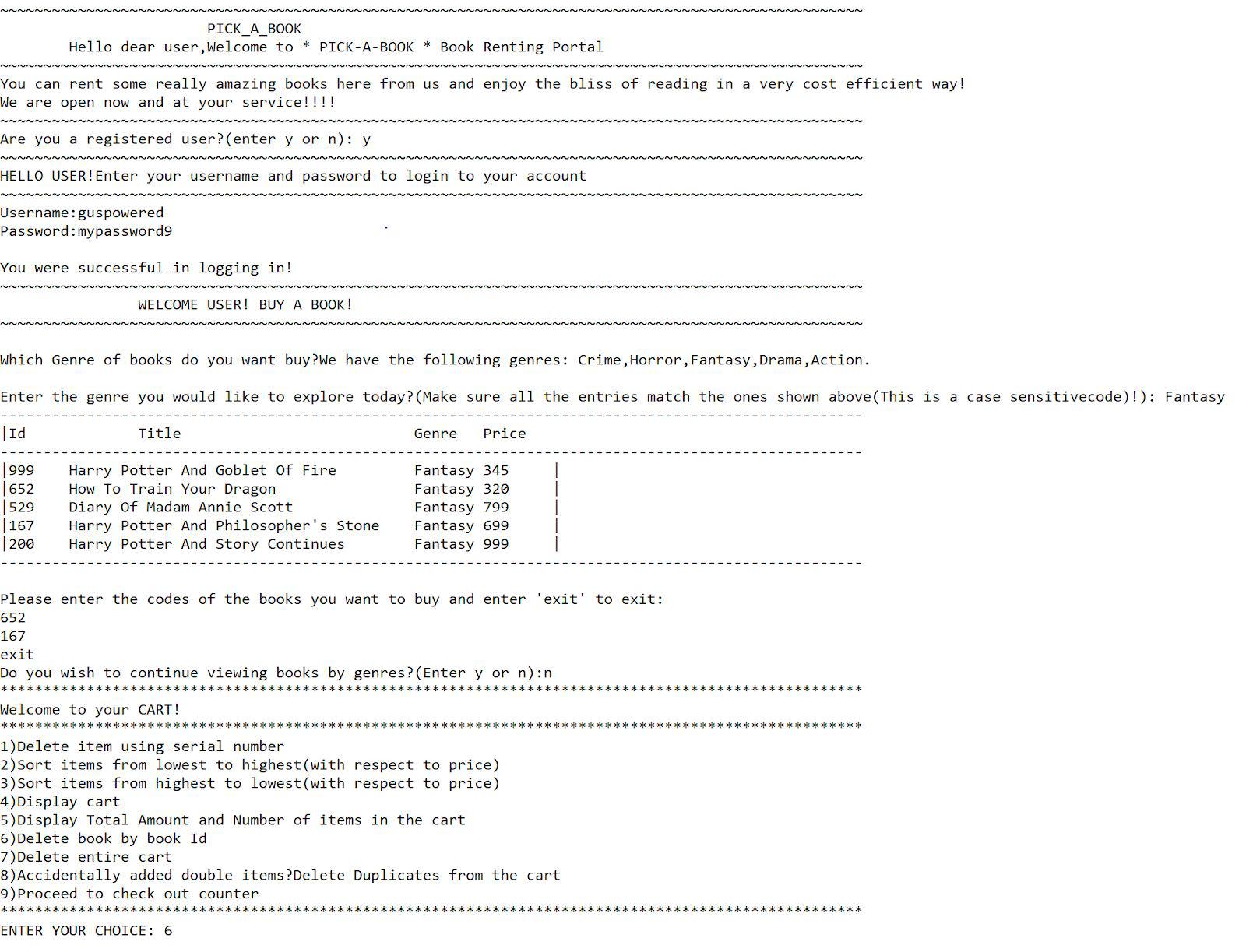


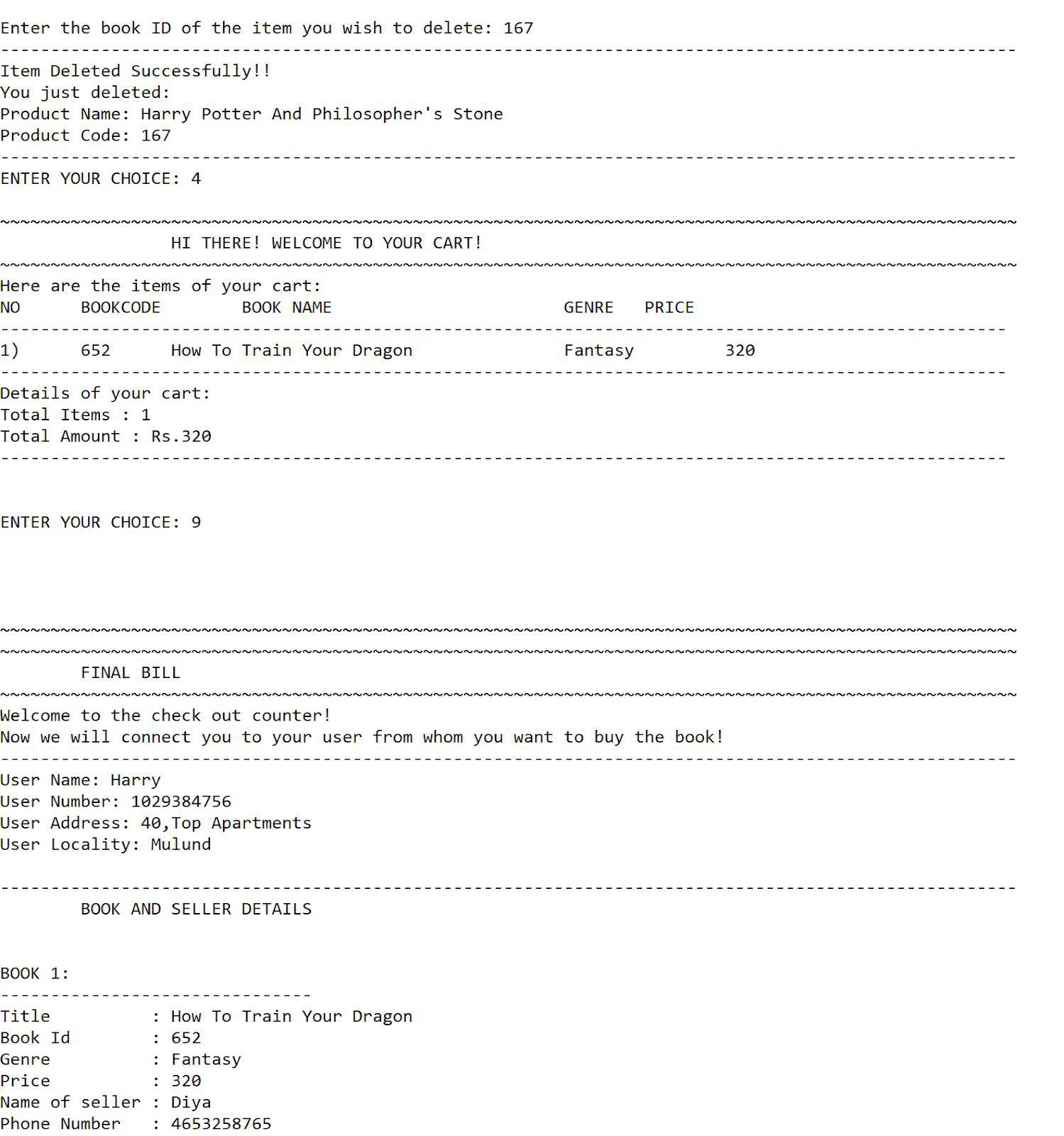
**OUTPUT CASE 5:**





**OUTPUT CASE 6:**

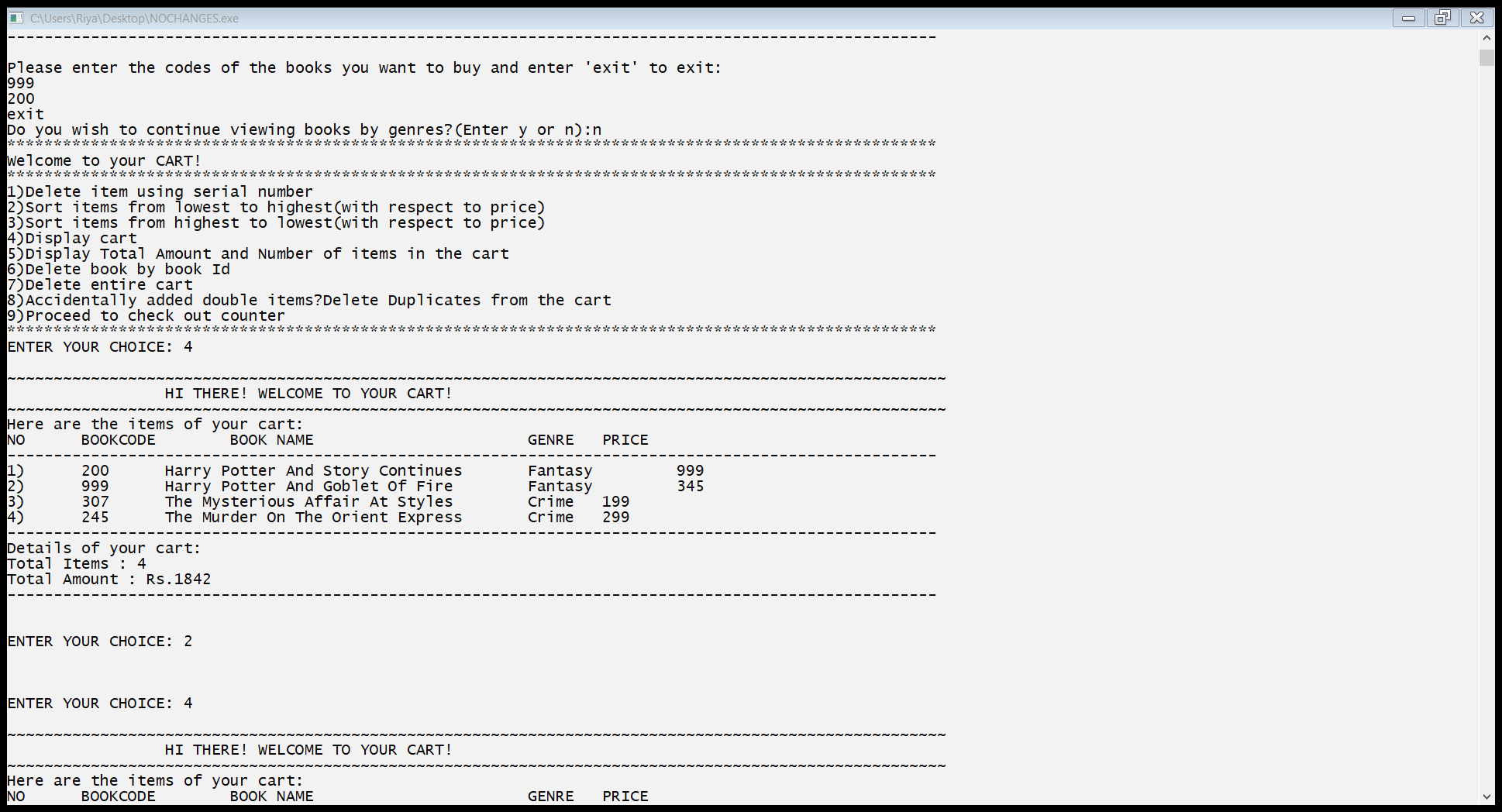
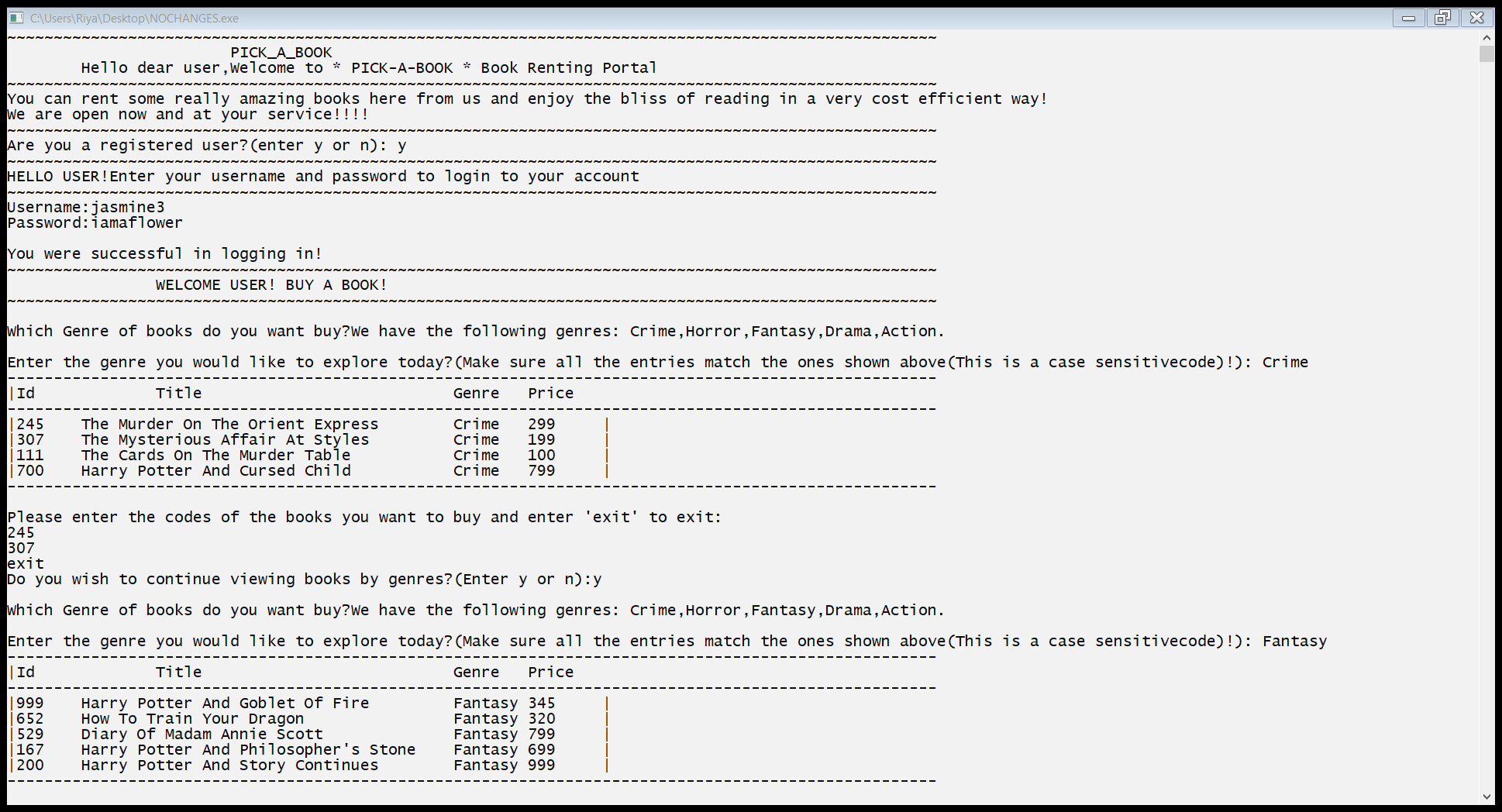
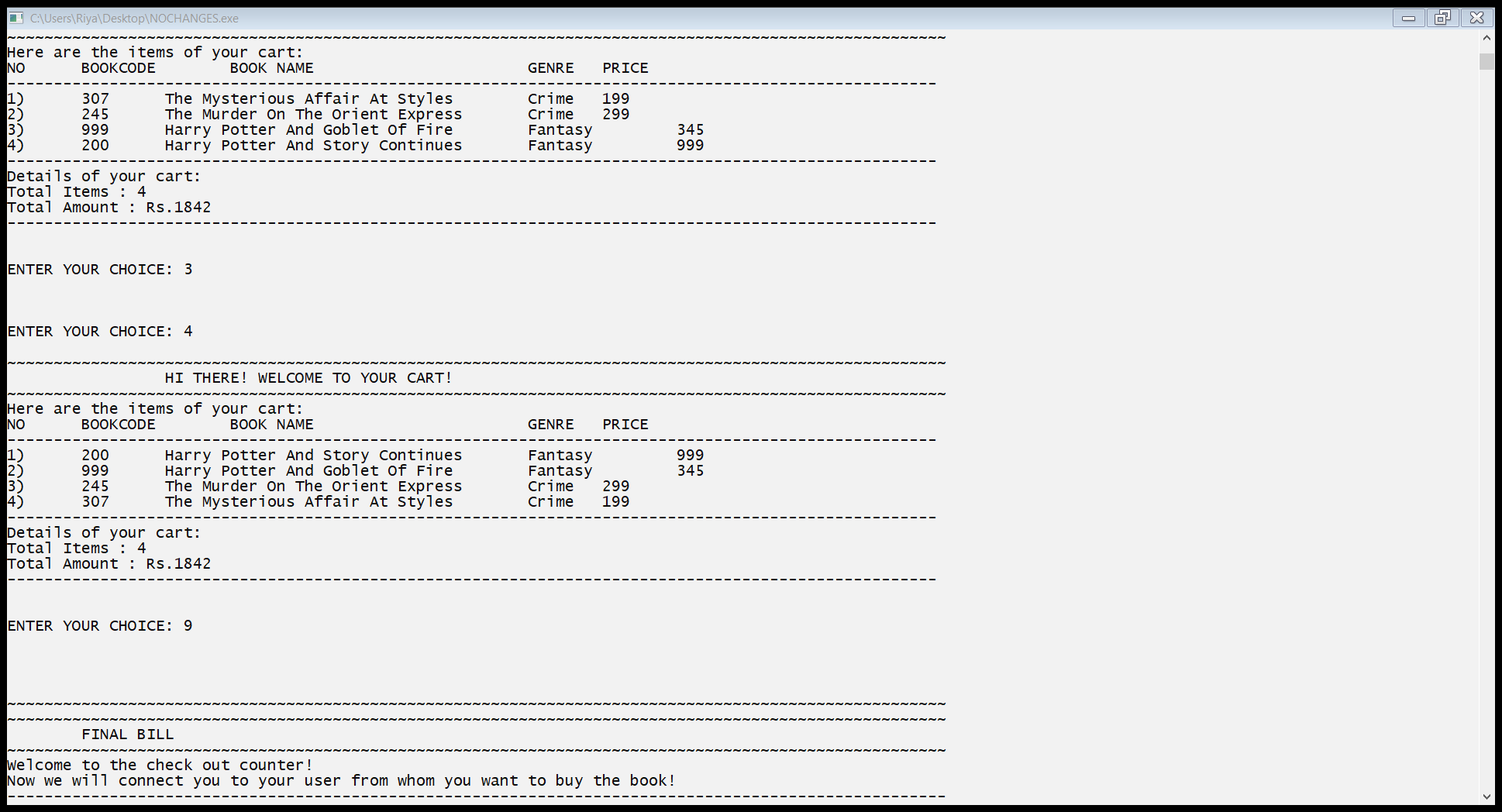


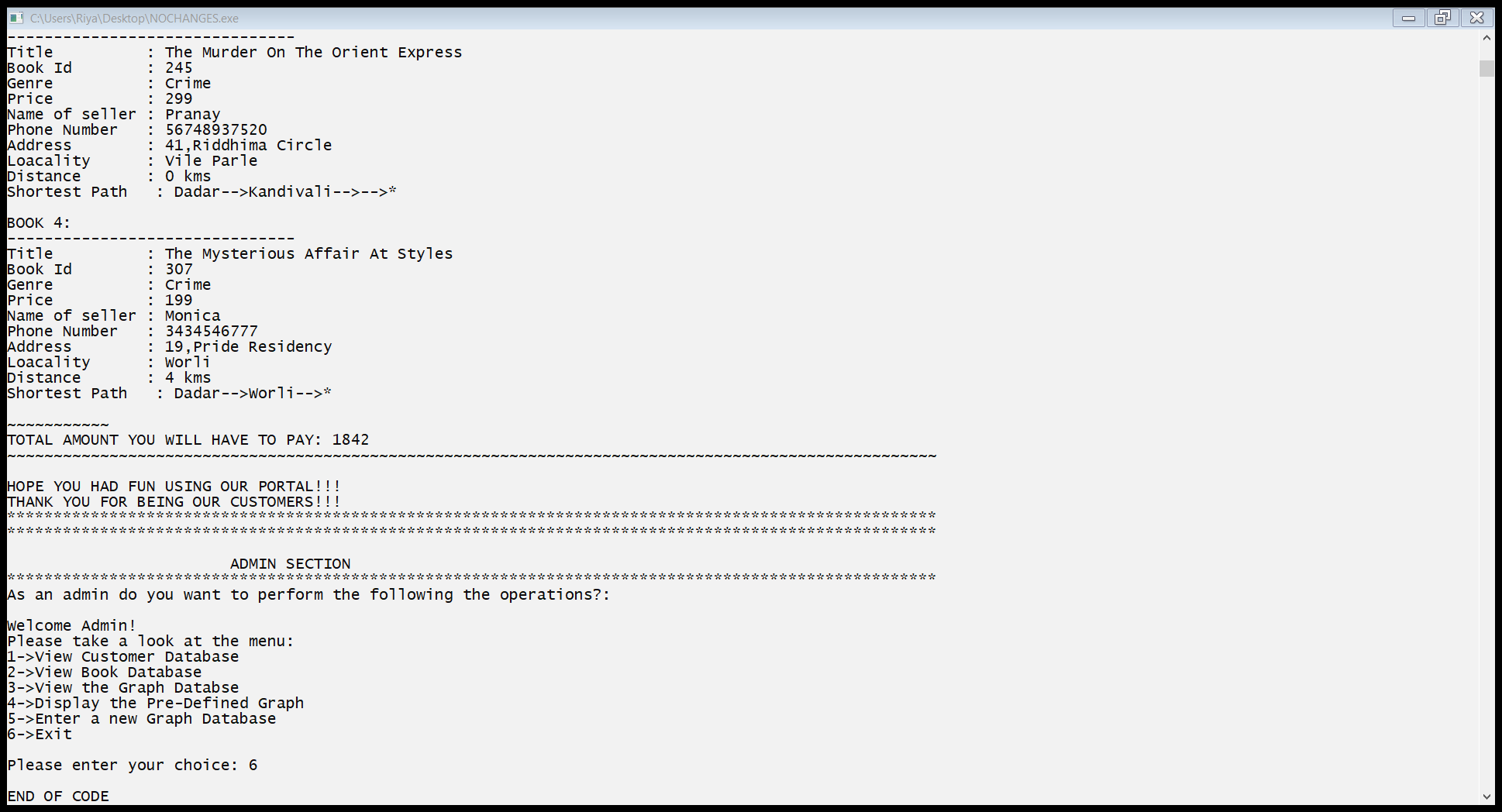
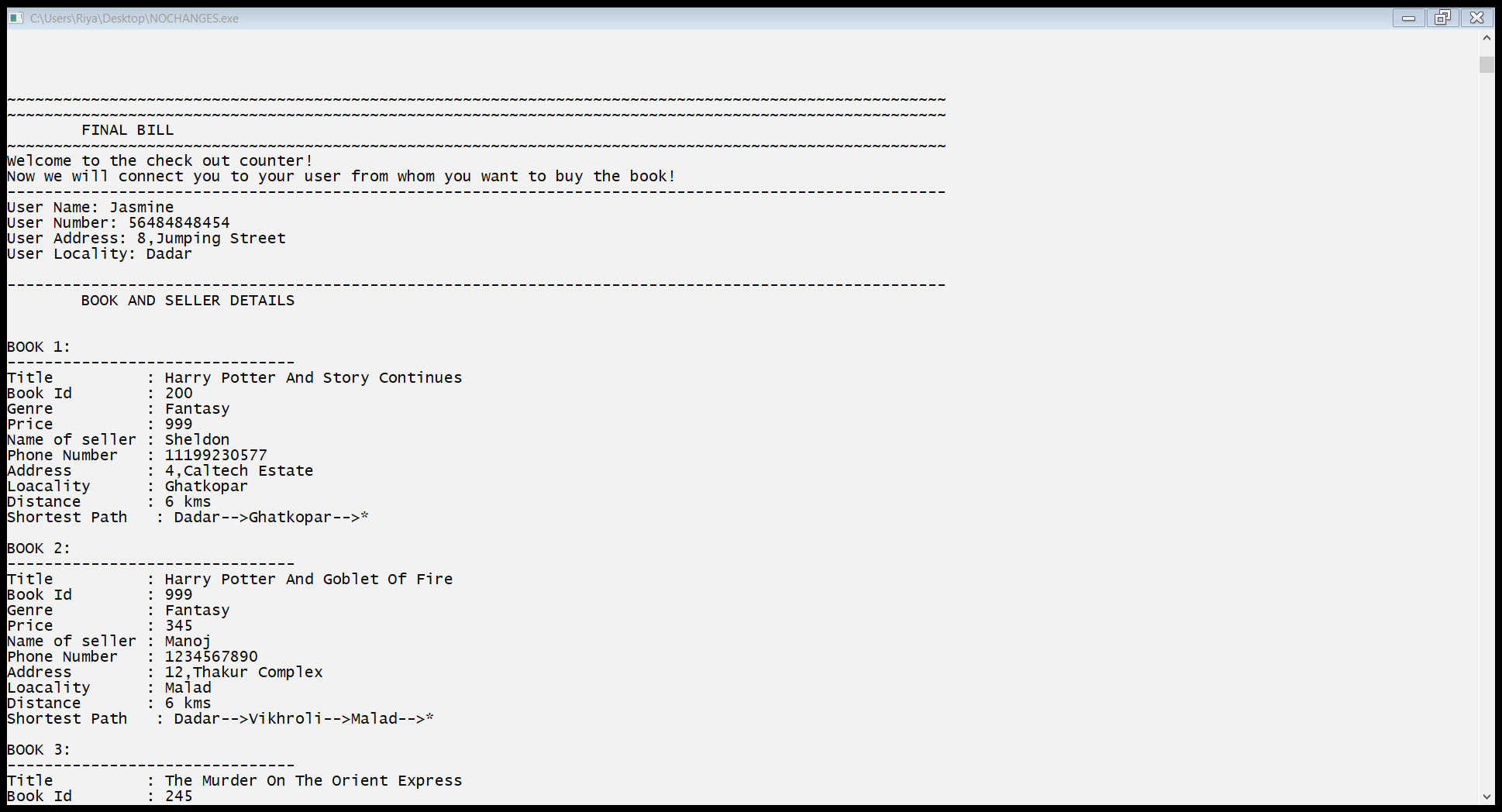






**OUTPUT CASE 7:**

** **

****

A.8 **Observations and learnings:**

From this we learn to implement the ADT of 3 major data structures: arrays,linked lists and graphs and we learn their implementation and functionalities.

Various data structures are being used in different applications in different ways all over the world. We tried to do the same. We used all the necessary ds to make our project a success. Important observations made were: every data structure has its importance. Even if one was missing from the world, things wouldn’t be a s simple as they are now.

A.9 **Conclusion:**

It is important to apply data structures properly in the applications otherwise the purpose isn’t met.  The use of all the ds used by us was needed in its place. It also ensured a huge amount of learning with respect to real life applications, pros and cons.

We are Absolutely grateful to Radhika ma’am for giving us this project where we explored and learned so much and also pushed ourselves to do the best.