**ASSIGNMENT 2**

**BIT 106**

**INTRODUCTION TO PROGRAMMING IN JAVA**

****

**Name: I Nyoman Surya Pradipta**

**Student ID: E1900344**

**Room.java**

*/\*\*  
 \* Room class defines a simple object type that represents a room.  
 \* @author I Nyoman Surya Pradipta  
 \* Student ID: E1900344  
 \* Date: 9 December 2020  
 \* Java version: 14.0.2  
 \* IDE : IntelliJ IDEA  
 \*/*public class Room {  
 private int numOfBeds;  
 private String guestName;  
 private boolean bookingStatus;  
 private double roomTariff;  
  
 */\*\*  
 \* The default constructor used to initialise the default value.  
 \*/* public Room(){  
 *// initialise default Room object* numOfBeds = 2;  
 guestName = "Nobody";  
 bookingStatus = false;  
 roomTariff = 100.00;  
 }  
  
 */\*\*  
 \* The setNumOfBeds setter method, which is used to set a new value NumOfBed variable.  
 \* @param newNumOfBed to assign the parameter value to the numOfBeds variable.  
 \* @return void No value to return.  
 \*/* public void setNumOfBeds(int newNumOfBed) {  
 if (newNumOfBed >= 1 && newNumOfBed <= 4) { *// inclusive 1-4* numOfBeds = newNumOfBed; *// assign new value* }  
 }  
  
 */\*\*  
 \* The setRoomTariff setter method, which is used to set a new value roomTariff variable.  
 \* @param newRoomTariff to assign the parameter value to the roomTariff variable.  
 \* @return void No value to return.  
 \*/* public void setRoomTariff(double newRoomTariff) {  
 if (newRoomTariff > 0) { *// value cannot be negative* roomTariff = newRoomTariff; *// assign new value* }  
 }  
  
 */\*\*  
 \* The bookRoom service method, which is used to set  
 \* a guestName and bookingStatus variables.  
 \* @param guestName to assign the parameter value to the guestName variable.  
 \* @return void No value to return.  
 \*/* public void bookRoom(String guestName) {  
 bookingStatus = true; *// room booked* this.guestName = guestName; *// assign new value* }  
  
 */\*\*  
 \* The vacateRoom service method, which is used to set  
 \* a guestName and bookingStatus variables to default value.  
 \* @return void No value to return.  
 \*/* public void vacateRoom() {  
 bookingStatus = false; *// not booked* guestName = "Nobody"; *// assign to default* }  
  
 */\*\*  
 \* The getNumOfBeds getter method, which is used to get the numOfBeds value.  
 \* @return int Value of numOfBeds.  
 \*/* public int getNumOfBeds() {  
 return numOfBeds;  
 }  
  
 */\*\*  
 \* The getRoomTariff getter method, which is used to get the roomTariff value.  
 \* @return double Value of roomTariff.  
 \*/* public double getRoomTariff() {  
 return roomTariff;  
 }  
  
 */\*\*  
 \* The getGuestName getter method, which is used to get the guestName value.  
 \* @return String Value of guestName.  
 \*/* public String getGuestName() {  
 return guestName;  
 }  
  
 */\*\*  
 \* The isBooked service method, which is used to get the bookingStatus value.  
 \* @return boolean Value of bookingStatus.  
 \*/* public boolean isBooked() {  
 return bookingStatus;  
 }  
  
 */\*\*  
 \* The toString query method, which is used to return the details information of the Room object.  
 \* @return String Details information of the Room object.  
 \*/* public String toString() {  
 return String.*format*("Room with %d beds, tariff %.2f, and guest named %s.", numOfBeds,roomTariff, guestName);  
 }  
}

**Hostel.java**

*/\*\*  
 \* The Hostel class defines an object which is  
 \* the container of the Room object.  
 \* @author I Nyoman Surya Pradipta  
 \* Student ID: E1900344  
 \* Date: 9 December 2020  
 \* Java version: 14.0.2  
 \* IDE : IntelliJ IDEA  
 \*/*public class Hostel {  
 *// declaration constant size of array* private final Room[] rooms;  
  
 */\*\*  
 \* A constructor with one argument,  
 \* which is used to initialise size of the Room array.  
 \* @param size accepts size array from client.  
 \*/* public Hostel(int size) {  
 if (size < 20 || size > 100) { *// invalid size* size = 50; *// default size* }  
 rooms = new Room[size + 1]; *// initialise array size  
 // invocation of private methods* addRoom(size);  
 setNumOfBed(size);  
 setRoomTariff(size);  
 }  
  
 */\*\*  
 \* The addRoom method is used to add new Room object to the Hostel.  
 \* @param size To get the size of array from constructor.  
 \* @return void No value to return.  
 \*/* private void addRoom(int size) {  
 for (int i = 0; i <= size; i++) { *// traverse array  
 // add object to array.* if (i == 0 ) {  
 rooms[i] = new Room();  
 }  
 if (i > 5 && i < (size - 5)) {  
 rooms[i] = new Room();  
 }  
 }  
 }  
  
 */\*\*  
 \* The setRoomTariff setter method, which is used to set the room tariff  
 \* room 0 to $1500.00, and all the even-numbered rooms to $150.00  
 \* @param size To get the size of array from constructor.  
 \* @return void No value to return.  
 \*/* private void setRoomTariff(int size) {  
 for (int i = 0; i <= size; i++) {  
 if ( i % 2 == 0) { *// even number* if (i == 0) { *// index 0 or room 0* rooms[i].setRoomTariff(1500.00);  
 }  
 else  
 rooms[i].setRoomTariff(150.00);  
 }  
 }  
 }  
  
 */\*\*  
 \* The setNumOfBed setter method, which is used to set new value of beds  
 \* from the last five rooms to 1, and rooms one through five to 4.  
 \* @param size To get the size of array from constructor.  
 \* @return void No value to return.  
 \*/* private void setNumOfBed(int size) {  
 for (int i = 0; i <= size; i++) {  
 if (i >= 1 && i <= 5) { *// rooms one through five* rooms[i] = new Room(); *// add object to array* rooms[i].setNumOfBeds(4); *// set the number of beds* }  
 if (i >= (size - 5) && i <= size) { *// the last five rooms* rooms[i] = new Room(); *// add object to array* rooms[i].setNumOfBeds(1); *// set the number of beds* }  
 }  
 }  
  
 */\*\*  
 \* The getNumOfBeds getter method, which is used to get  
 \* the reference the Room object.  
 \* @param roomNumber The value representing a room number.  
 \* @return reference Room object or null.  
 \*/* public Room getRoom(int roomNumber) {  
 for (int i = 0; i < rooms.length; i++) {  
 if (roomNumber == i) {  
 return rooms[i]; *// reference Room object* }  
 }  
 return null; *// invalid room number* }  
  
 */\*\*  
 \* The getNumOfBeds getter method, which is used to  
 \* get number of rooms which are booked.  
 \* @return int The number of room which are booked.  
 \*/* public int numOfBookedRooms() {  
 int bookedRoom = 0;  
 for (Room room : rooms) { *// traverse collection of rooms* if (room.isBooked()) { *// check boolean conditions* bookedRoom++; *// num booked room increased by 1* }  
 }  
 return bookedRoom;  
 }  
  
 */\*\*  
 \* The numOfVacantRooms getter method, which is used to get  
 \* the number of rooms not booked.  
 \* @return int The number of rooms not booked.  
 \*/* public int numOfVacantRooms() {  
 int vacateRoom = 0;  
 for (Room room : rooms) { *// traverse collection of rooms* if (!room.isBooked()) { *// check boolean conditions* vacateRoom++; *// num booked room increased by 1* }  
 }  
 return vacateRoom;  
 }  
  
 */\*\*  
 \* The totalTariff getter method, which is used to get and calculate  
 \* total tariff from all the booked rooms.  
 \* @return double Total tariff from all the booked rooms.  
 \*/* public double totalTariff() {  
 double income = 0.0;  
 for (Room room : rooms) { *// traverse collection of rooms* if(room.isBooked()) *// check boolean conditions* income += room.getRoomTariff(); *// add room tariff to income* }  
 return income;  
  
 }  
  
 */\*\*  
 \* The getAvailableRooms getter method, which is used to get a list of  
 \* all vacant rooms that have sufficient beds for potential guests.  
 \* @param numOfGuest Representing the number of guests that needed the room.  
 \* @return String A list of all vacant rooms.  
 \*/* public String getAvailableRooms(int numOfGuest) {  
 StringBuilder vacantRooms = new StringBuilder(); *// create StringBuilder object* for (int i = 0; i < rooms.length; i++) {  
 if (numOfGuest == rooms[i].getNumOfBeds()) {  
 vacantRooms.append("\n(Room number ").append(i).append(") ").append(rooms[i].toString());  
 }  
 }  
 return vacantRooms.toString();  
 }  
  
 */\*\*  
 \* The findGuestRoomNumber getter method, which is used to  
 \* search for the name of the first guest in the hostel.  
 \* @param guestName Representing a guest's name.  
 \* @return int Room number when found  
 \*/* public int findGuestRoomNumber(String guestName) {  
 int roomNum = -1;  
 for (int i = 0; i < rooms.length; i++) {  
 if (rooms[i].getGuestName().equalsIgnoreCase(guestName)) {  
 return i;  
 }  
 }  
 return roomNum;  
 }  
}

**HostelMain.java**

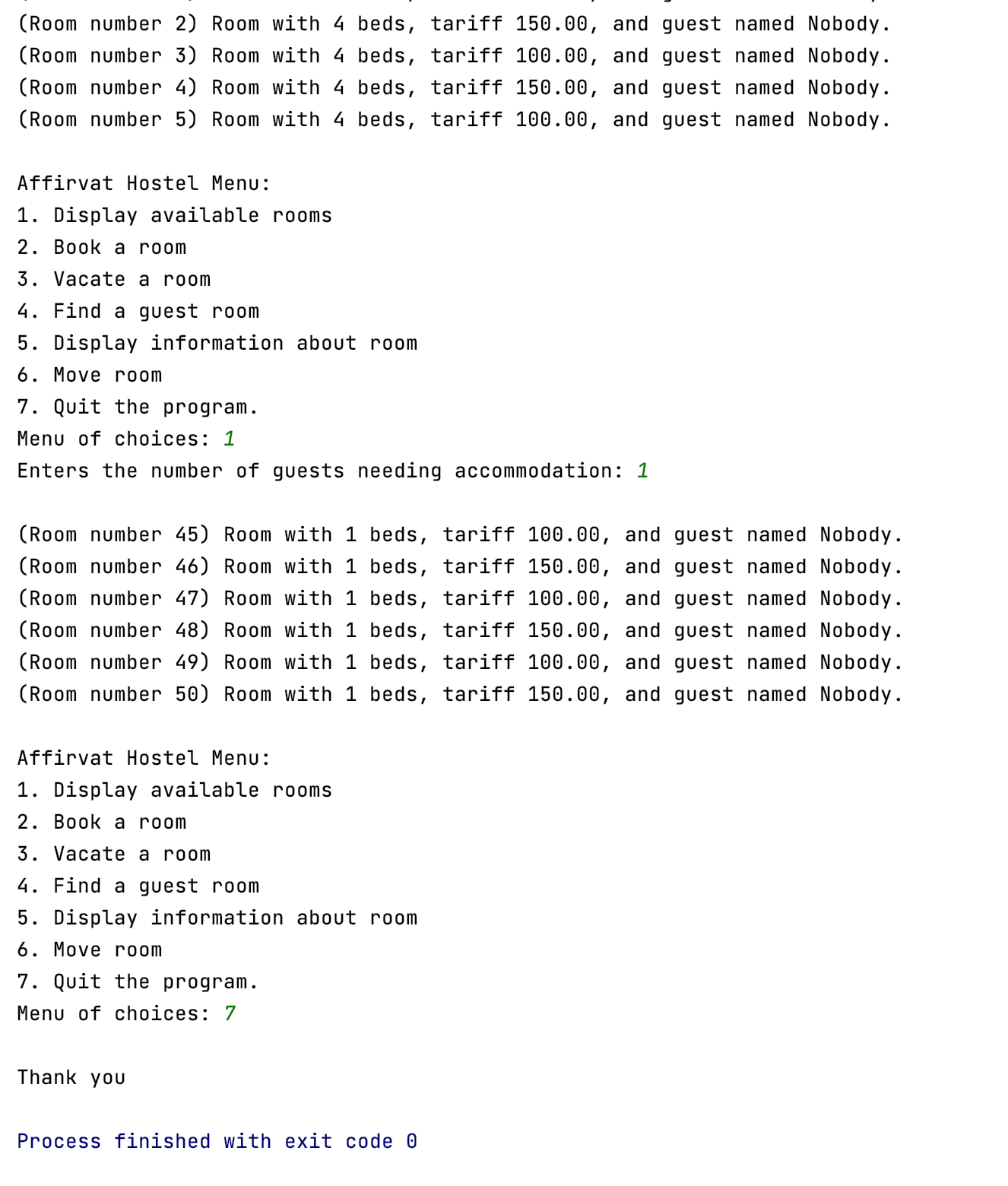
import java.util.Scanner;  
  
*/\*\*  
 \* The HostelMain program implements an application consisting of various menus:  
 \* display available rooms, book rooms, vacate rooms,  
 \* find a guest room, display information about room,  
 \* and move room.  
 \* @author I Nyoman Surya Pradipta  
 \* Student ID: E1900344  
 \* Date: 9 December 2020  
 \* Java version: 14.0.2  
 \* IDE : IntelliJ IDEA  
 \*/*public class HostelMain {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Welcome to Affirvat Hostel\nHow many rooms are available today: ");  
 int size = input.nextInt();  
 Hostel Affirvat = new Hostel(size);  
 int menu = 0;  
 while (menu != 7) {  
 System.out.println("\nAffirvat Hostel Menu:\n1. Display available rooms");  
 System.out.println("2. Book a room");  
 System.out.println("3. Vacate a room");  
 System.out.println("4. Find a guest room");  
 System.out.println("5. Display information about room");  
 System.out.println("6. Move room");  
 System.out.println("7. Quit the program.");  
 System.out.print("Menu of choices: ");  
 menu = input.nextInt();  
 input.nextLine();  
 switch (menu) {  
 case 1 -> {  
 System.out.print("Enters the number of guests needing accommodation: ");  
 int numOfGuest = input.nextInt();  
 System.out.println(Affirvat.getAvailableRooms(numOfGuest));  
 }  
 case 2 -> {  
 System.out.print("Enters name of the guest: ");  
 String guestName = input.nextLine();  
 System.out.print("Enter the available room number to book: ");  
 int numOfBookRoom = input.nextInt();  
 Affirvat.getRoom(numOfBookRoom).bookRoom(guestName);  
 }  
 case 3 -> {  
 System.out.print("Enters the number of the room to be vacated: ");  
 int numOfVacateRoom = input.nextInt();  
 if (Affirvat.getRoom(numOfVacateRoom).isBooked()) {  
 Affirvat.getRoom(numOfVacateRoom).vacateRoom();  
 } else {  
 System.out.println("\nInvalid data, room not booked yet");  
 }  
 }  
 case 4 -> {  
 System.out.print("Enters a guest's name: ");  
 String guestName = input.nextLine();  
 if (Affirvat.findGuestRoomNumber(guestName) != -1) {  
 System.out.print("\nGuests are in the room number ");  
 System.out.println(Affirvat.findGuestRoomNumber(guestName));  
 }  
 else  
 System.*out*.println("\nInvalid data, name not found");  
 }  
 case 5 -> {  
 System.*out*.println("\nThe number of booked rooms: " + Affirvat.numOfBookedRooms());  
 System.*out*.println("The number of empty rooms: " + Affirvat.numOfVacantRooms());  
 System.*out*.printf("The total tariff of all booked rooms $%.2f \n", Affirvat.totalTariff());  
 }  
 case 6 -> {  
 System.out.print("Enter the room number: ");  
 int roomNumber = input.nextInt();  
 if (Affirvat.getRoom(roomNumber).isBooked()) {  
 Affirvat.getRoom(roomNumber).vacateRoom();  
 System.out.println("The room has been vacated");  
 System.out.print("Enter the room number where the guest wants to move: ");  
 int roomMoved = input.nextInt();  
 input.nextLine();  
 while (Affirvat.getRoom(roomMoved).isBooked()) {  
 System.out.print("Room has booked, enter another room: ");  
 roomMoved = input.nextInt();  
 input.nextLine();  
 }  
 System.out.print("Enters a guest's name: ");  
 String guestName = input.nextLine();  
 Affirvat.getRoom(roomMoved).bookRoom(guestName);  
 System.out.println("Room successfully moved");  
 }  
 else {  
 System.out.println("\nInvalid data, room is not booked");  
 }  
 }  
 case 7 -> System.out.println("\nThank you");  
 }  
 }  
 }  
}

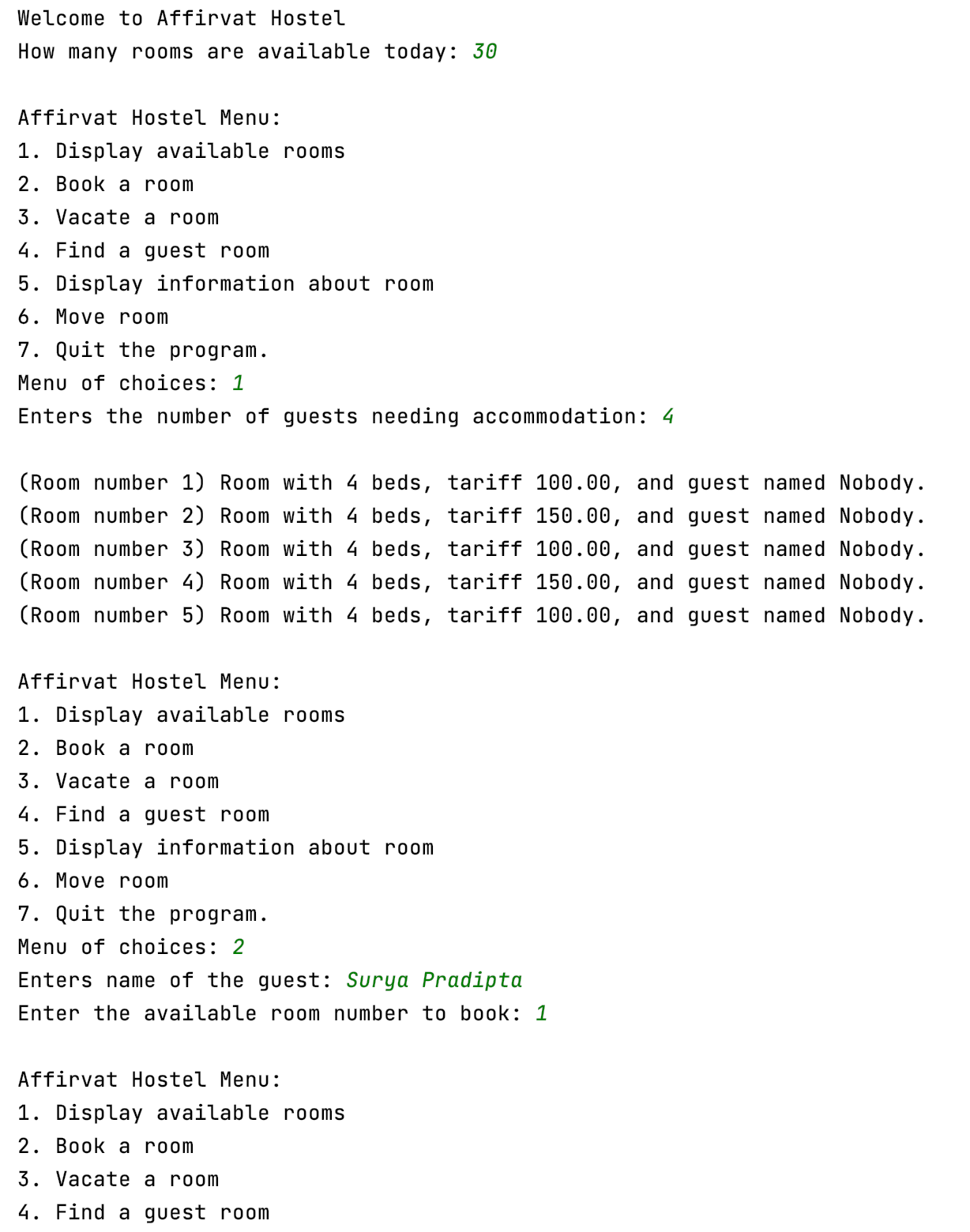
**Sample Output**

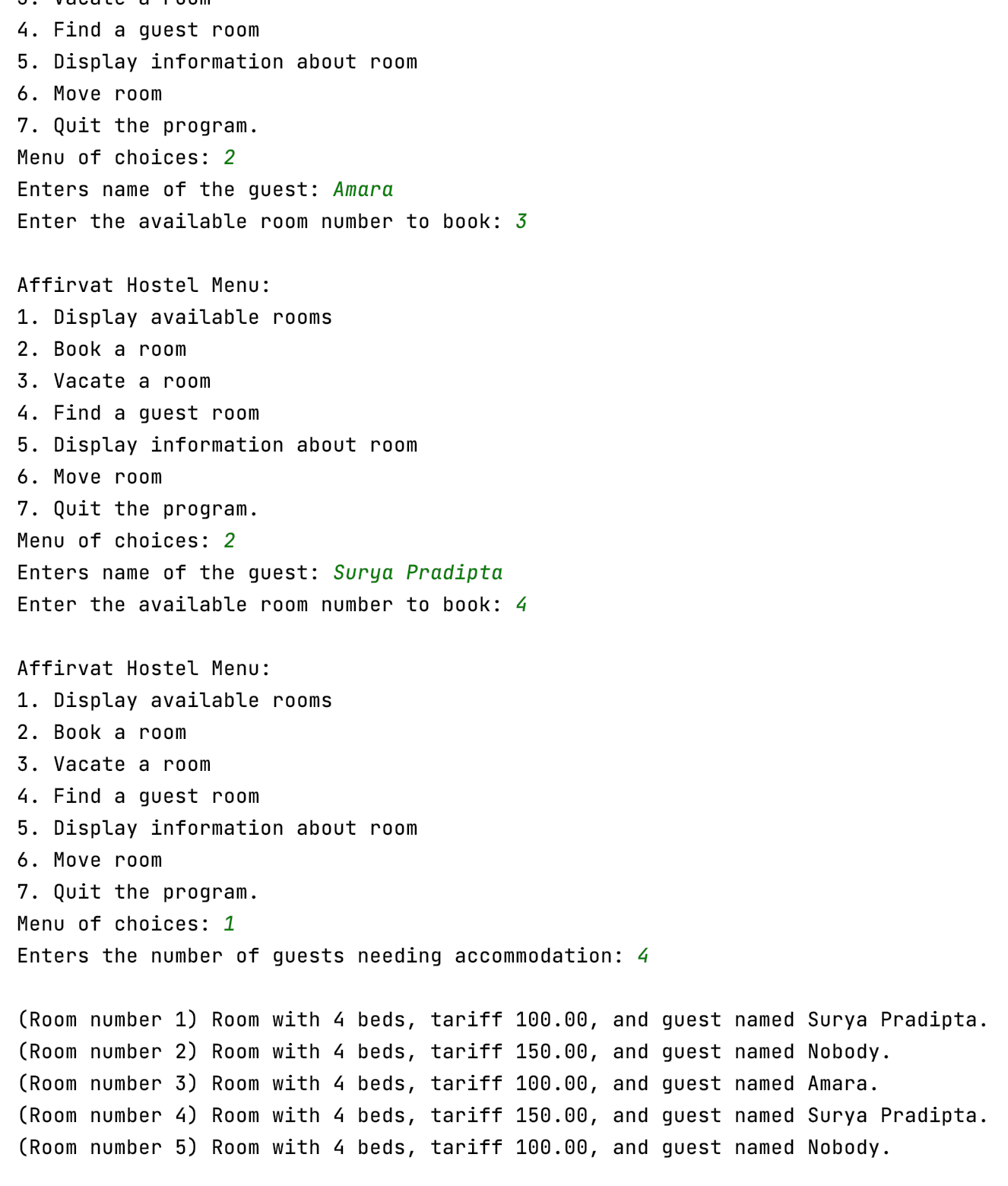
****

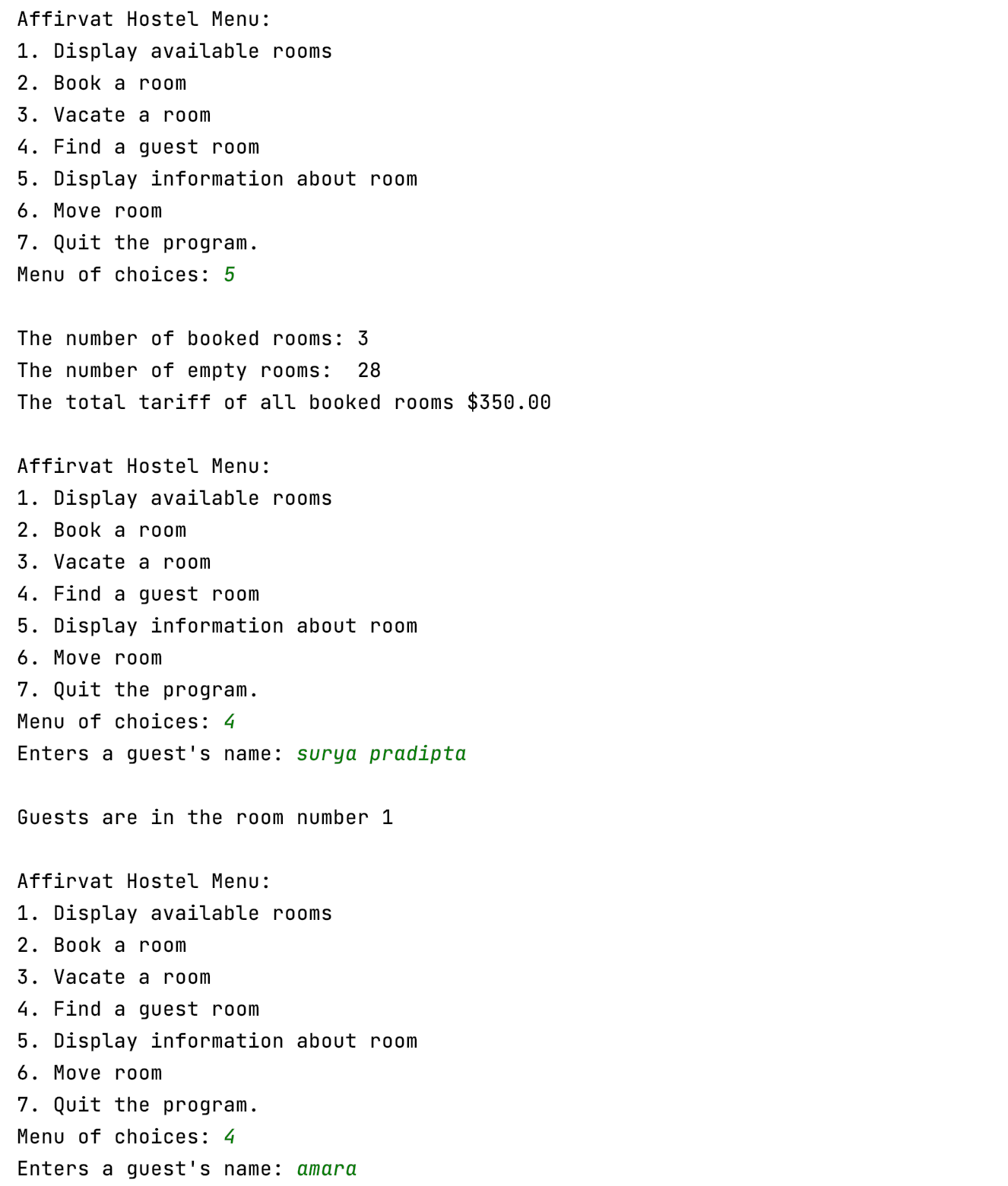
****

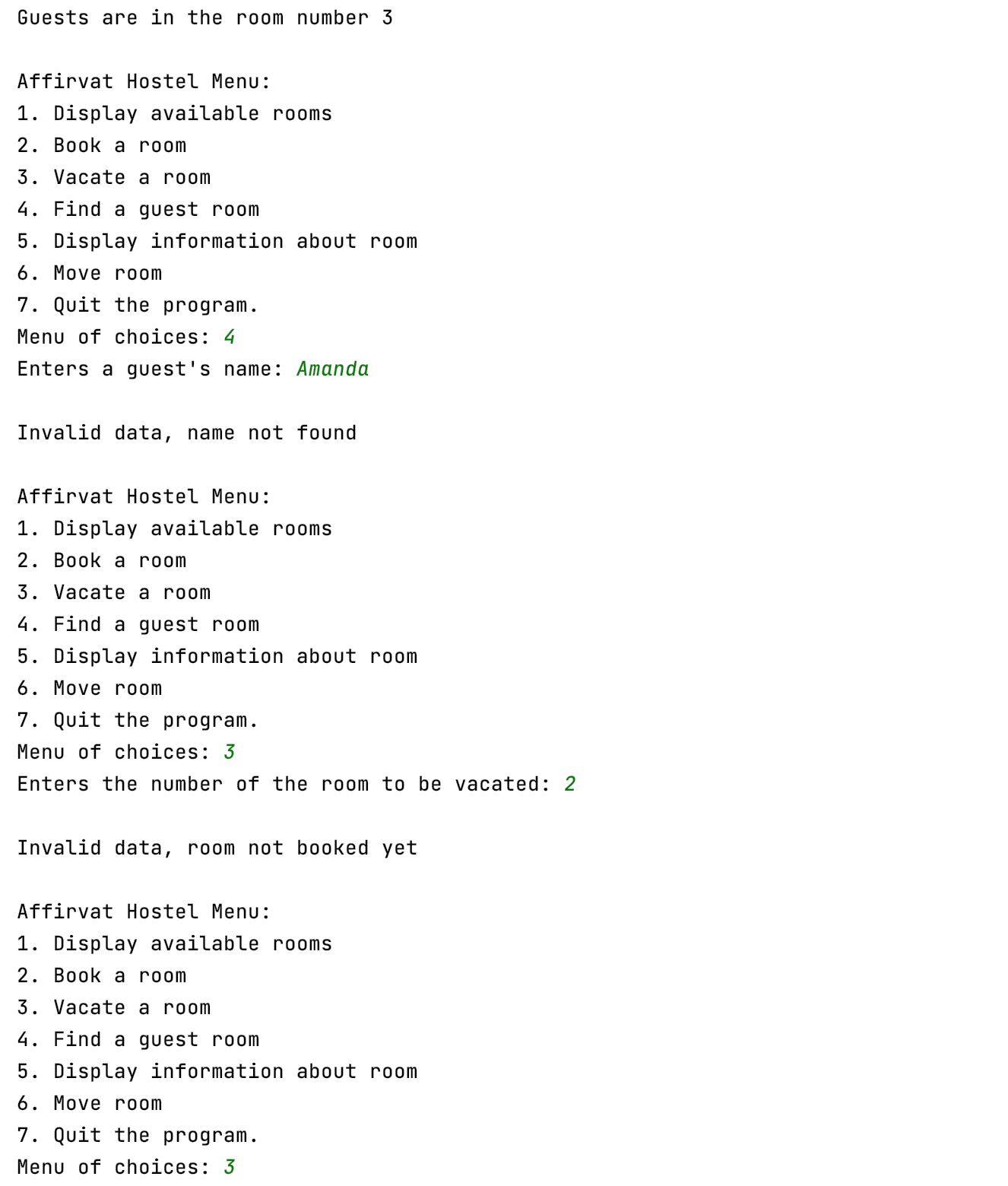
****

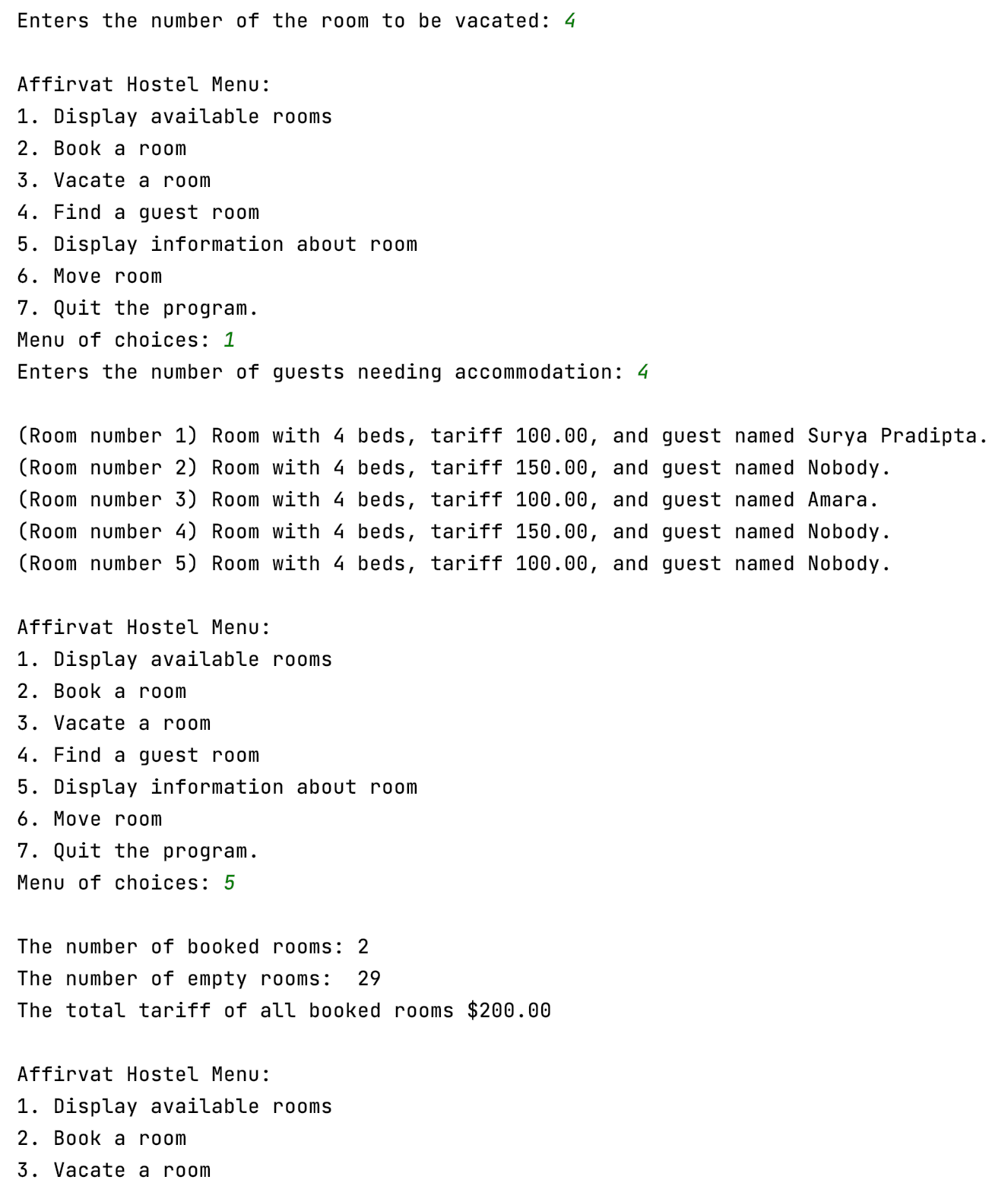
****

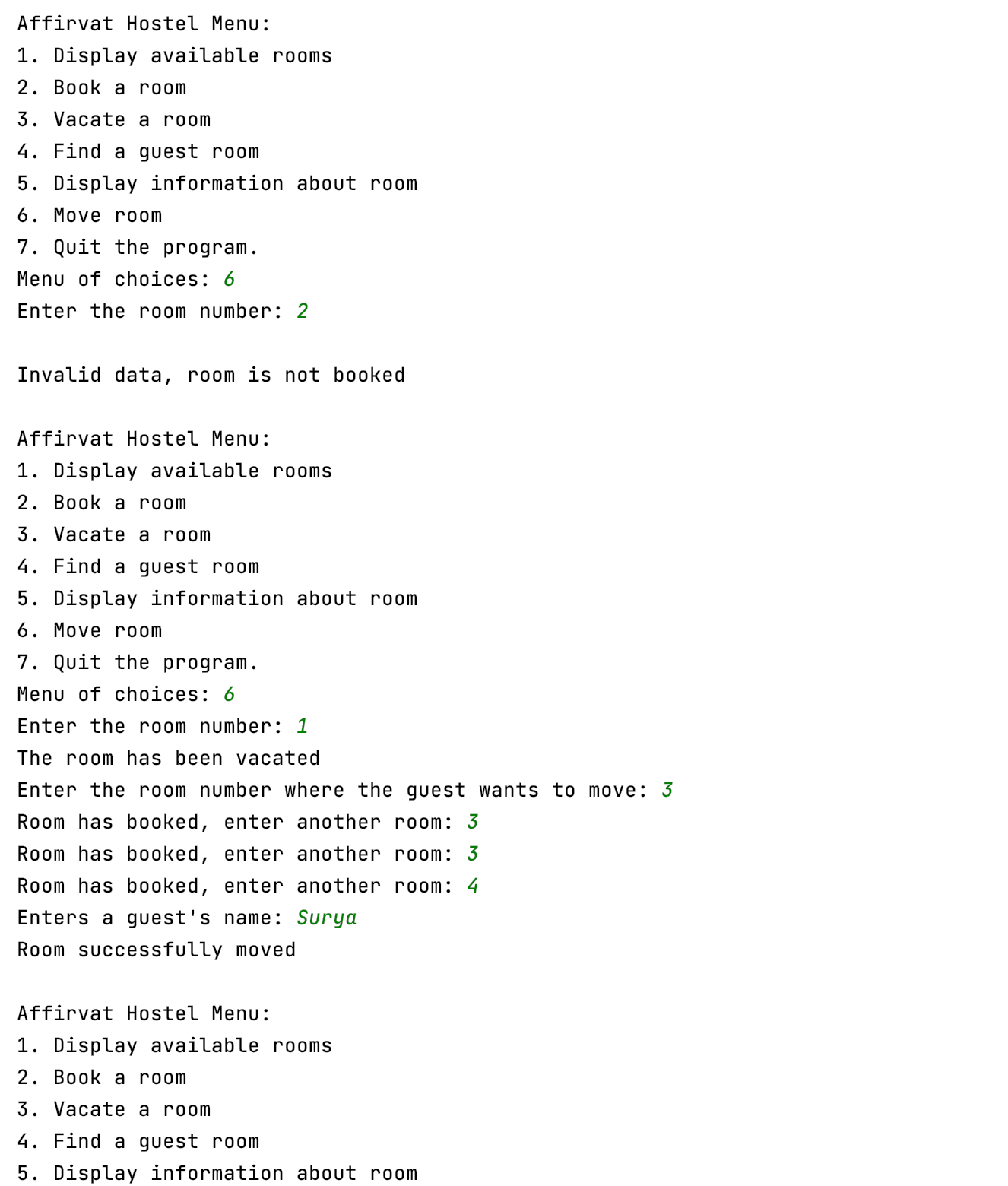
****

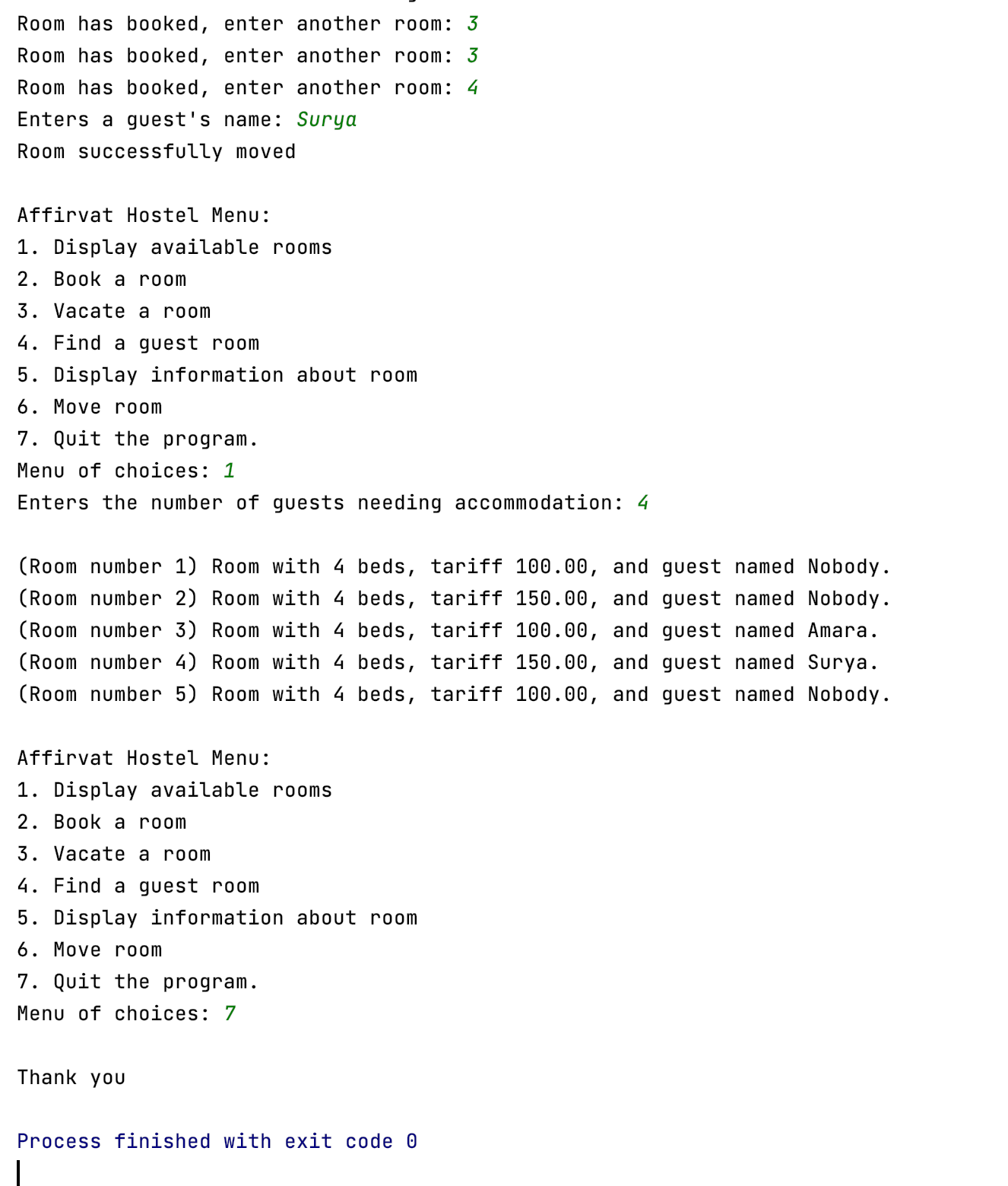
****

****

****

****

****

****

****