GROUP MEMBERS OOP PROJECT

EIHAB KHAN 221751

ABDULLAH KHAN 221748

EDEN JOSEPH 221658

SUBMITTED TO MAM MARIAM

DATE 21 MAY 2023

CEP

MTS 2A

COMPLEX ENGINEERING PROBLEM

UNIVERSITY MANAGEMENT SYSTEM USING C ++

INTRODUCTION

In this complex engineering problem me and my teammates were asked to build a functioning university management system which included multiple aspects such as teacher's portal gaming zone library university cafeteria transport system etc

Each consisting of task that are to performed with the knowledge we possessed by attending all the oops lab from the begining of the second semester of mechatronics under the teaching skills of Mam Mariam

Implementation

In order to explain each part properly during the presentation we have displayed separate programs made but each of the group members . At the end of the report we have compiled a single long program to present an exact result.

TRANSPORT PROGRAM

```
#include <iostream>

#include <string>


using namespace std;


struct Route {

    int routeId;

    string departure;

    string arrival;
    string busNumber;
    string departureTime;
    int seatsAvailable;
};
const int MAX_ROUTES = 100;
Route routes[MAX_ROUTES];
int numRoutes = 0;
const int MAX_RESERVATIONS = 100;
int reservations[MAX_RESERVATIONS];
int numReservations = 0;
int lastReservation = -1;

void addRoute(int routeId, string departure, string arrival, string busNumber, string departureTime,
int seatsAvailable) {
    if (numRoutes < MAX_ROUTES) {
        Route route;
        route.routeId = routeId;
        route.departure = departure;
        route.arrival = arrival;
        route.busNumber = busNumber;
        route.departureTime = departureTime;
        route.seatsAvailable = seatsAvailable;

        routes[numRoutes++] = route;
        cout << "Route added successfully!" << endl;
    } else {
        cout << "Maximum number of routes reached." << endl;
    }
}
```

```
void showRoutes() {
    cout << "Available Routes:" << endl;
    for (int i = 0; i < numRoutes; i++) {
        cout << "Route ID: " << routes[i].routeId << endl;
        cout << "Departure: " << routes[i].departure << endl;
        cout << "Arrival: " << routes[i].arrival << endl;
        cout << "Departure Time: " << routes[i].departureTime << endl;
        cout << "Bus Number: " << routes[i].busNumber << endl;
        cout << "Seats Available: " << routes[i].seatsAvailable << endl;
        cout << endl;
    }
}

bool reserveSeat(int routeId) {
    for (int i = 0; i < numRoutes; i++) {
        if (routes[i].routeId == routeId) {
            if (routes[i].seatsAvailable > 0) {
                routes[i].seatsAvailable--;
                lastReservation = routeId;
                reservations[numReservations++] = routeId;
                cout << "Seat reserved successfully!" << endl;
                cout << "Route ID: " << routes[i].routeId << endl;
                cout << "Departure: " << routes[i].departure << endl;
                cout << "Arrival: " << routes[i].arrival << endl;
                cout << "Departure Time: " << routes[i].departureTime << endl;
                cout << "Bus Number: " << routes[i].busNumber << endl;
                return true;
            } else {
                cout << "No seats available for this route." << endl;
                return false;
            }
        }
    }
    cout << "Invalid Route ID." << endl;
    return false;
}

void showLastReservation() {
    if (lastReservation != -1) {
        cout << "Last Reservation Details:" << endl;
        for (int i = 0; i < numRoutes; i++) {
            if (routes[i].routeId == lastReservation) {
                cout << "Route ID: " << routes[i].routeId << endl;
                cout << "Departure: " << routes[i].departure << endl;
                cout << "Arrival: " << routes[i].arrival << endl;
                cout << "Departure Time: " << routes[i].departureTime << endl;
                cout << "Bus Number: " << routes[i].busNumber << endl;
                return;
            }
        }
    }
    cout << "No previous reservations." << endl;
}

void checkAvailableBuses() {
    cout << "Number of available buses: " << numRoutes << endl;
}
```

```
int main() {
    // Sample routes
    addRoute(1, "University", "Murree Road", "BUS001", "17:30", 10);
    addRoute(2, "University", "Islamabad Sectors", "BUS002", "16:300", 5);
    addRoute(3, "University", "Rawalpindi Cantt", "BUS003", "17:00", 3);

    int choice;

    do {
        cout << "Transport System Menu:" << endl;
        cout << "1. Show Routes" << endl;
        cout << "2. Reserve a Seat" << endl;
        cout << "3. Show Last Reservation" << endl;
        cout << "4. Check Available Buses" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
cout << endl ;
        switch (choice) {
            case 1:
                showRoutes();
                cout << endl ;
                break;
            case 2: {
                int routeId;
                cout << "Enter the Route ID: ";
                cin >> routeId;
                reserveSeat(routeId);
                cout << endl ;
                break;
            }
            case 3:
                showLastReservation();
                cout << endl ;
                break;
            case 4:
                checkAvailableBuses();
                cout << endl ;
                break;
            case 5:
                cout << "Exiting..." << endl;
                cout << endl ;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
                cout << endl ;
                break;
        }

        cout << endl;
    } while (choice != 5);

    return 0;
}
```

UNIVERSITY CAFÉ PROGRAM

```
#include <iostream>
#include <string>
#include <sstream>
```

```
#include <ctime>

using namespace std;

// Structure to hold menu item information
struct MenuItem {
    string name;
    double price;
};

// Function to display the menu for a specific day
void displayMenu(const MenuItem menu[], int size) {
    cout << "Menu for the day : " << endl;
    for (int i = 0; i < size; i++) {
        cout << i + 1 << ". " << menu[i].name << " - Rs " << menu[i].price << endl;
    }
    cout << endl;
}

// Function to calculate the total bill including 4% GST
double calculateTotalBill(const MenuItem order[], int size) {
    double total = 0.0;
    for (int i = 0; i < size; i++) {
        total += order[i].price;
    }
    double gst = total * 0.04;
    total += gst;
    return total;
}
```

```
int main() {
   // Define menus for each day
   MenuItem menus[7][3] = {
      {{"Roll", 40}, {"Fruit Chat", 200}, {"Cake", 100}},
      {{"Shawarma", 170}, {"Macaroni", 200}, {"AppleJuice", 70}},
      {{"Zinger Shawarma", 200}, {"Samosa", 40}, {"Peach Juice", 70}},
      {{"Fries", 100}, {"Pizza Slice", 200}, {"Grapes Juice", 70}},
      {{"Lays ", 60}, {"Burger", 200}, {"Tea", 60}},
      {{"Pasta", 100}, {"Grilled Wrap", 300}, {"Strawberry Milkshake", 200}},
      {{"Noodles", 70}, {"Loaded Fries ", 200}, {"Slush", 100}}
   };

   // Get current day of the week
   time_t now = time(NULL);
   tm* currentDate = localtime(&now);
   int currentDay = currentDate->tm_wday;

   // Get the menu for the current day
   MenuItem* currentMenu = menus[currentDay];
   int menuSize = sizeof(menus[currentDay]) / sizeof(menus[currentDay][0]);

   MenuItem order[menuSize];
   int orderSize = 0;
   string choice;

   // Display the menu for the current day
   displayMenu(currentMenu, menuSize);
```

```
    do {
        cout << "Choose an item number to add to your order (or enter 'Bill' to show the bill): ";
        cin >> choice;

        if (choice == "Bill") {
            break;
        } else {
            int itemIndex;
            istringstream iss(choice);
            if (iss >> itemIndex && itemIndex >= 1 && itemIndex <= menuSize) {
                order[orderSize++] = currentMenu[itemIndex - 1];
                cout << "Item added to your order." << endl;
            } else {
                cout << "Invalid choice. Please try again." << endl;
            }
        }
    } while (true);
    cout << endl ;
    // Calculate and display the total bill
    double totalBill = calculateTotalBill(order, orderSize);
    cout << "Your Order :" << endl;
    for (int i = 0; i < orderSize; i++) {
        cout << order[i].name << " - Rs " << order[i].price << endl;
    }
    cout << "Total Price : Rs " << totalBill << endl;

    return 0;
}
```

GAMING ZONE PROGRAM

```
#include <iostream>

#include <vector>

using namespace std;


// Function to display the Tic-Tac-Toe board

void displayBoard(const vector<vector<char> >& board) {

  for (int row = 0; row < 3; row++) {

    for (int col = 0; col < 3; col++) {

      cout << board[row][col] << " ";

    }

    cout << endl;

  }

}


// Function to check if a player has won the game

bool checkWin(const vector<vector<char> >& board, char player) {

  // Check rows

  for (int row = 0; row < 3; row++) {

    if (board[row][0] == player && board[row][1] == player && board[row][2] == player)

      return true;

  }


  // Check columns

  for (int col = 0; col < 3; col++) {

    if (board[0][col] == player && board[1][col] == player && board[2][col] == player)

      return true;

  }
```

```
    // Check diagonals
    if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||
        (board[0][2] == player && board[1][1] == player && board[2][0] == player))
        return true;

    return false;
}

// Function to check if the game is a draw
bool checkDraw(const vector<vector<char> >& board) {
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            if (board[row][col] == '#')
                return false; // Empty cell found, game is not a draw
        }
    }
    return true; // All cells are filled, game is a draw
}

// Function to play the Tic-Tac-Toe game
void playTicTacToe() {
    vector<vector<char> > board(3, vector<char>(3, '#'));
    int row, col;
    char currentPlayer = 'X';

    cout << "Tic-Tac-Toe Game" << endl;
cout << "Player 1: X" << endl;
cout << "Player 2: O" << endl;
cout << endl;

while (true) {
  displayBoard(board);

  cout << "Player " << currentPlayer << "'s turn. Enter row (0-2): ";
  cin >> row;
  cout << "Enter column (0-2): ";
  cin >> col;

  if (row < 0 || row > 2 || col < 0 || col > 2) {
    cout << "Invalid position. Try again." << endl;
    continue;
  }

  if (board[row][col] != '#') {
    cout << "Position already occupied. Try again." << endl;
    continue;
  }

  board[row][col] = currentPlayer;

  if (checkWin(board, currentPlayer)) {
    displayBoard(board);
    cout << "Player " << currentPlayer << " wins!" << endl;
```

```
            break;
        }

        if (checkDraw(board)) {
            displayBoard(board);
            cout << "Game is a draw!" << endl;
            break;
        }

        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
        cout << endl;
    }
}

int main() {
    playTicTacToe();
    return 0;
}
```

LIBRARY PROGRAM

```
#include <iostream>
#include <string>
#include <vector>
#include <ctime>

using namespace std;

class Book {
```

```
private:
    string title;
    string author;
    int id;
    bool available;
    time_t dueDate;

public:
    Book(string _title, string _author, int _id) {
        title = _title;
        author = _author;
        id = _id;
        available = true;
        dueDate = 0;
    }

    string getTitle() const {
        return title;
    }

    string getAuthor() const {
        return author;
    }

    int getID() const {
        return id;
    }

    bool isAvailable() const {
        return available;
    }

    time_t getDueDate() const {
        return dueDate;
    }

    void setAvailability(bool _available) {
        available = _available;
    }

    void setDueDate(time_t _dueDate) {
        dueDate = _dueDate;
    }
};

class Student {
private:
    string name;
    int rollNumber;
    Book* issuedBook;

public:
    Student(string _name, int _rollNumber) {
        name = _name;
        rollNumber = _rollNumber;
        issuedBook = NULL;
    }
    string getName() const {
        return name;
    }

    int getRollNumber() const {
        return rollNumber;
    }

    Book* getIssuedBook() const {
        return issuedBook;
    }

    void issueBook(Book* book, time_t dueDate) {
        issuedBook = book;
        issuedBook->setAvailability(false);
        issuedBook->setDueDate(dueDate);
    }

    void returnBook() {
        if (issuedBook != NULL) {
            issuedBook->setAvailability(true);
            issuedBook->setDueDate(0);
            issuedBook = NULL;
        }
    }
};

class Library {
private:
    vector<Book*> books;
    vector<Student*> students;

public:
    void addBook(string title, string author, int id) {
        Book* newBook = new Book(title, author, id);
        books.push_back(newBook);
        cout << "Book added: " << title << " (ID: " << id << ")" << endl;
    }

    void addStudent(string name, int rollNumber) {
        Student* newStudent = new Student(name, rollNumber);
        students.push_back(newStudent);
        cout << "Student added: " << name << " (Roll Number: " << rollNumber << ")" << endl;
    }

    Book* findBookByID(int id) {
        for (int i = 0; i < books.size(); i++) {
            if (books[i]->getID() == id) {
                return books[i];
            }
        }
        return NULL;
    }

    Student* findStudentByRollNumber(int rollNumber) {
        for (int i = 0; i < students.size(); i++) {
            if (students[i]->getRollNumber() == rollNumber) {
                return students[i];
            }
        }
        return NULL;
    }

    void issueBookToStudent(int bookID, int rollNumber) {
        Book* book = findBookByID(bookID);
        Student* student = findStudentByRollNumber(rollNumber);

        if (book == NULL) {
            cout << "Book not found with ID: " << bookID << endl;
            return;
        }

        if (student == NULL) {
            cout << "Student not found with Roll Number: " << rollNumber << endl;
            return;
        }

        if (!book->isAvailable()) {
            cout << "Book is already issued to another student." << endl;
            return;
        }

        if (student->getIssuedBook() != NULL) {
            cout << "Student has already issued a book. Return the previous book before issuing a new
one." << endl;
            return;
        }
        time_t currentTime = time(NULL);
        time_t dueDate = currentTime + (7 * 24 * 60 * 60); // Set due date as 7 days from current time

        student->issueBook(book, dueDate);
        cout << "Book '" << book->getTitle() << "' (ID: " << book->getID() << ") issued to student '" <<
student->getName() << "' (Roll Number: " << student->getRollNumber() << ")." << endl;
    }

    void returnBookFromStudent(int rollNumber) {
        Student* student = findStudentByRollNumber(rollNumber);

        if (student == NULL) {
            cout << "Student not found with Roll Number: " << rollNumber << endl;
            return;
        }

        Book* issuedBook = student->getIssuedBook();
        if (issuedBook == NULL) {
            cout << "Student has not issued any book." << endl;
            return;
        }

        time_t currentTime = time(NULL);
        time_t dueDate = issuedBook->getDueDate();
        double daysLate = difftime(currentTime, dueDate) / (24 * 60 * 60);
        double fine = 0.0;

        if (daysLate > 0) {
            fine = daysLate * 10.0; // Assuming fine of 10 currency units per day of delay
        }

        cout << "Book '" << issuedBook->getTitle() << "' (ID: " << issuedBook->getID() << ") returned by
student '" << student->getName() << "' (Roll Number: " << student->getRollNumber() << ").";
        cout << " Fine: " << fine << " currency units." << endl;

        student->returnBook();
    }
};

int main() {
    Library library;

    // Add books to the library
    library.addBook("Book 1", "Author 1", 1);
    library.addBook("Book 2", "Author 2", 2);
    library.addBook("Book 3", "Author 3", 3);

    // Add students to the library
    library.addStudent("Student 1", 101);
    library.addStudent("Student 2", 102);
    library.addStudent("Student 3", 103);

    // Issue books to students
    library.issueBookToStudent(1, 101);
    library.issueBookToStudent(2, 102);
    library.issueBookToStudent(3, 103);
    // Return books from students
    library.returnBookFromStudent(101);
    library.returnBookFromStudent(102);

    return 0;
}
```

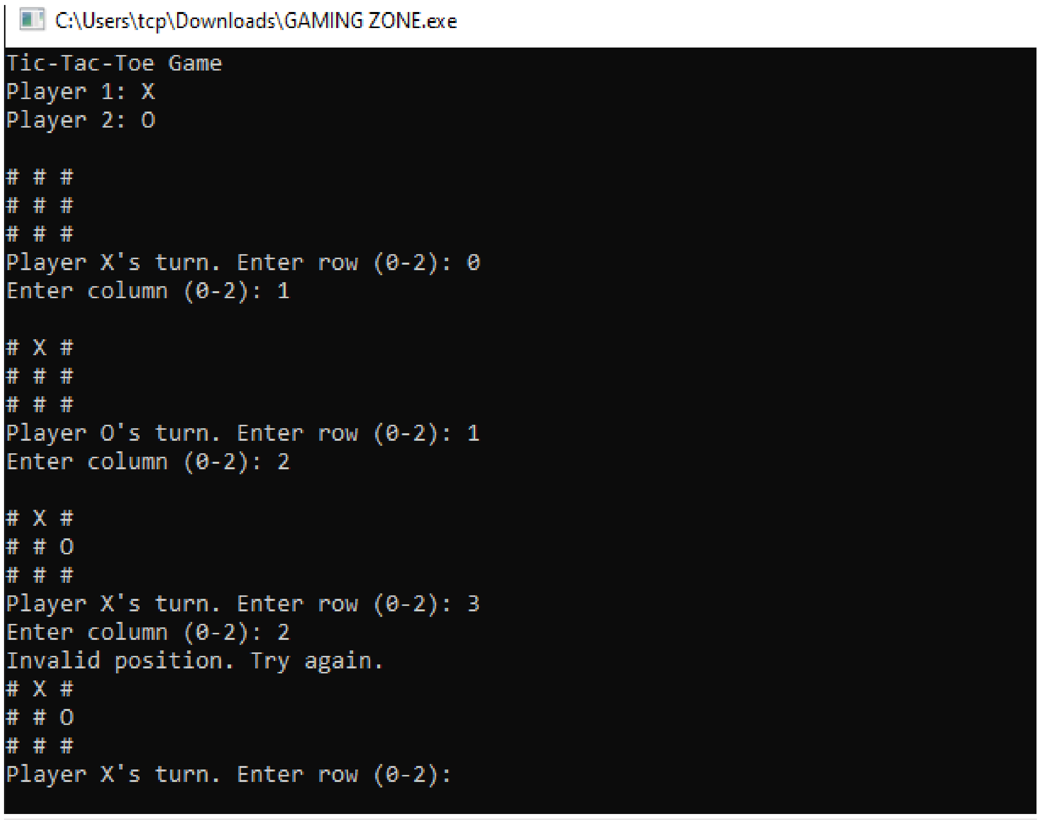
TEACHER PORTAL PROGRAM

```
#include <iostream>
#include <string>
using namespace std;

class Teacher {
protected:
    string classes[3];
public:
    void enterClass(string classNum[], int subjectNum) {
        for (int i = 0; i < 3; i++) {
            cout << "Enter your class " << i + 1 << " with name: ";
            cin >> classNum[i];
        }

        int c;
        cout << "Enter the class number you want to select: ";
        cin >> c;
        if (c >= 1 && c <= 3) {
```

```
        cout << "You have selected class " << classNum[c-1] << endl;
    }
    else {
        cout << "Invalid class selection." << endl;
    }

    string subjects[5];
    for (int j = 0; j < subjectNum; j++) {
        cout << "Enter subject " << j + 1 << " name: ";
        cin >> subjects[j];
    }

    int a;
    for (int j = 0; j < subjectNum; j++) {
        cout << "Press " << j + 1 << " for " << subjects[j] << endl;
    }
    cin >> a;
    if (a >= 1 && a <= subjectNum) {
        cout << "You have chosen the subject " << subjects[a-1] << endl;
    }
    else {
        cout << "Invalid subject selection." << endl;
    }
  }

  void students(string name[], int rollno[]) {
    int n;
    cout << "How many students do you want to enter? ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter the name and roll number for student " << i + 1 << ": ";
        cin >> name[i] >> rollno[i];
    }

    int m;
    cout << "Which student do you want to enter the marks for? Press 1 for the first and so on: ";
    cin >> m;
    if (m >= 1 && m <= n) {
        cout << "You are entering the marks for " << name[m-1] << " with roll no " << rollno[m-1] << endl;
    }
    else {
        cout << "Invalid student selection." << endl;
    }
  }

void marking() {
    cout << "You are entering marks for lab assessment" << endl;

    const int MAX_STUDENTS = 5; // Maximum number of students
    const int MAX_LAB_REPORTS = 5; // Maximum number of lab reports

    int marks[MAX_STUDENTS][MAX_LAB_REPORTS];
    double averages[MAX_STUDENTS];
    int totals[MAX_STUDENTS];

    int numStudents;
    cout << "Enter the number of students: ";
    cin >> numStudents;

    int numLabReports;
    cout << "Enter the number of lab reports: ";
    cin >> numLabReports;

    // Input marks for each student and lab report
    for (int i = 0; i < numStudents; i++) {
        cout << "Enter marks for Student " << i + 1 << endl;
        for (int j = 0; j < numLabReports; j++) {
            cout << "Lab Report " << j + 1 << ": ";
            cin >> marks[i][j];
        }
    }

    // Calculate average marks and total marks for each student
    for (int i = 0; i < numStudents; i++) {
        int sum = 0;
        for (int j = 0; j < numLabReports; j++) {
            sum += marks[i][j];
        }
        averages[i] = static_cast<double>(sum) / numLabReports;
        totals[i] = sum;
    }

    // Display average marks and total marks for each student
    cout << "Average Marks and Total Marks:" << endl;
    for (int i = 0; i < numStudents; i++) {
      cout << "Student " << i + 1 << ": Average=" << averages[i] << ", Total=" << totals[i] <<
endl;
    }
  }
};

int main() {
  Teacher t;
  string classNum[3];
  int subjectNum;
  string name[5]; // Assuming a maximum of 5 students
  int rollno[5]; // Assuming a maximum of 5 students

  t.enterClass(classNum, subjectNum);
  t.students(name, rollno);
  t.marking();

  return 0;
}
```

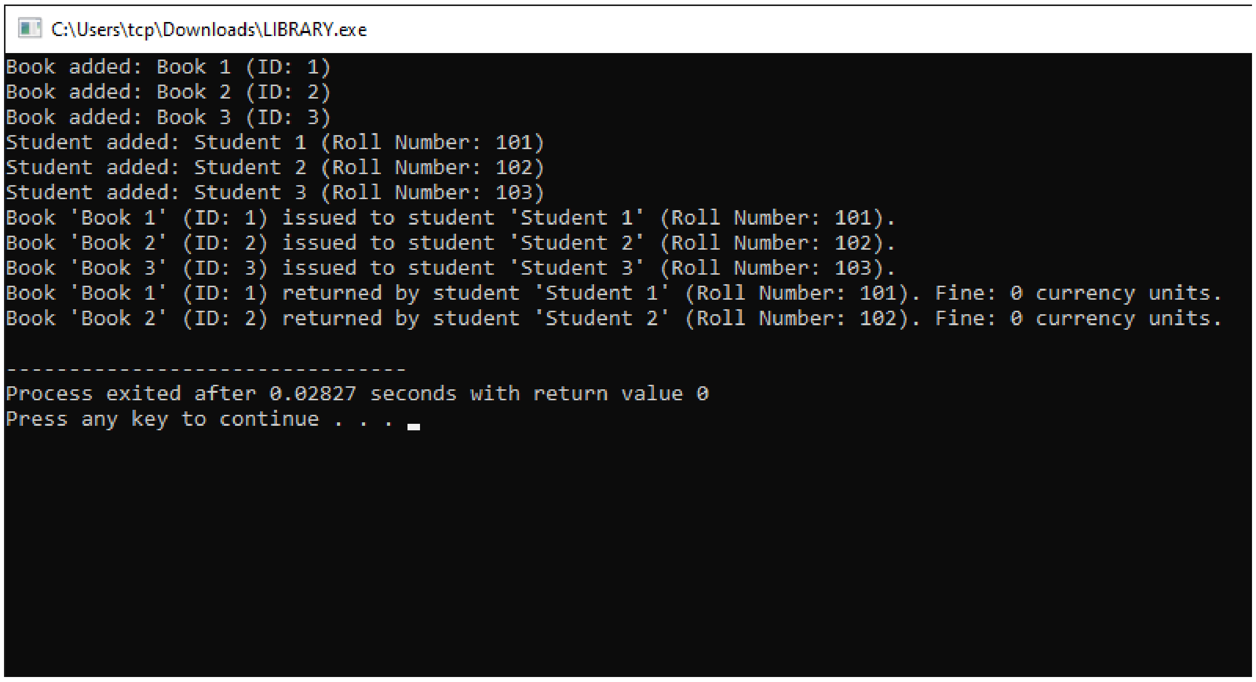
PROGRAM OUTPUTS

Gaming output



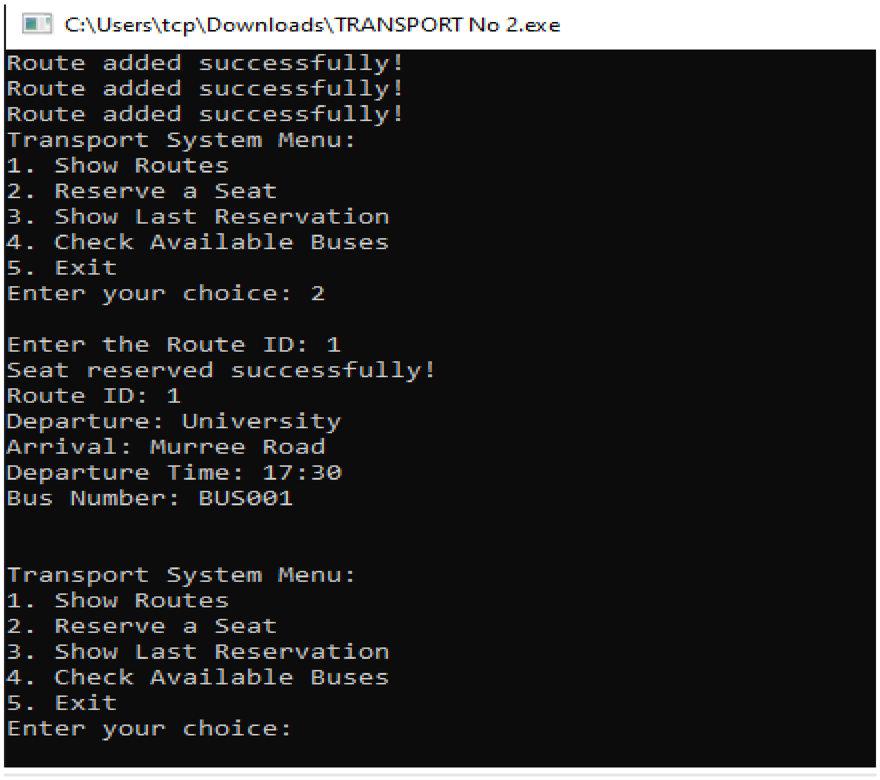
```
Tic-Tac-Toe Game
Player 1: X
Player 2: O

# # #
# # #
# # #
Player X's turn. Enter row (0-2): 0
Enter column (0-2): 1

# X #
# # #
# # #
Player O's turn. Enter row (0-2): 1
Enter column (0-2): 2

# X #
# # O
# # #
Player X's turn. Enter row (0-2): 3
Enter column (0-2): 2
Invalid position. Try again.
# X #
# # O
# # #
Player X's turn. Enter row (0-2):
```

Library output



```
Book added: Book 1 (ID: 1)
Book added: Book 2 (ID: 2)
Book added: Book 3 (ID: 3)
Student added: Student 1 (Roll Number: 101)
Student added: Student 2 (Roll Number: 102)
Student added: Student 3 (Roll Number: 103)
Book 'Book 1' (ID: 1) issued to student 'Student 1' (Roll Number: 101).
Book 'Book 2' (ID: 2) issued to student 'Student 2' (Roll Number: 102).
Book 'Book 3' (ID: 3) issued to student 'Student 3' (Roll Number: 103).
Book 'Book 1' (ID: 1) returned by student 'Student 1' (Roll Number: 101). Fine: 0 currency units.
Book 'Book 2' (ID: 2) returned by student 'Student 2' (Roll Number: 102). Fine: 0 currency units.

--------------------------------
Process exited after 0.02827 seconds with return value 0
Press any key to continue . . .
```

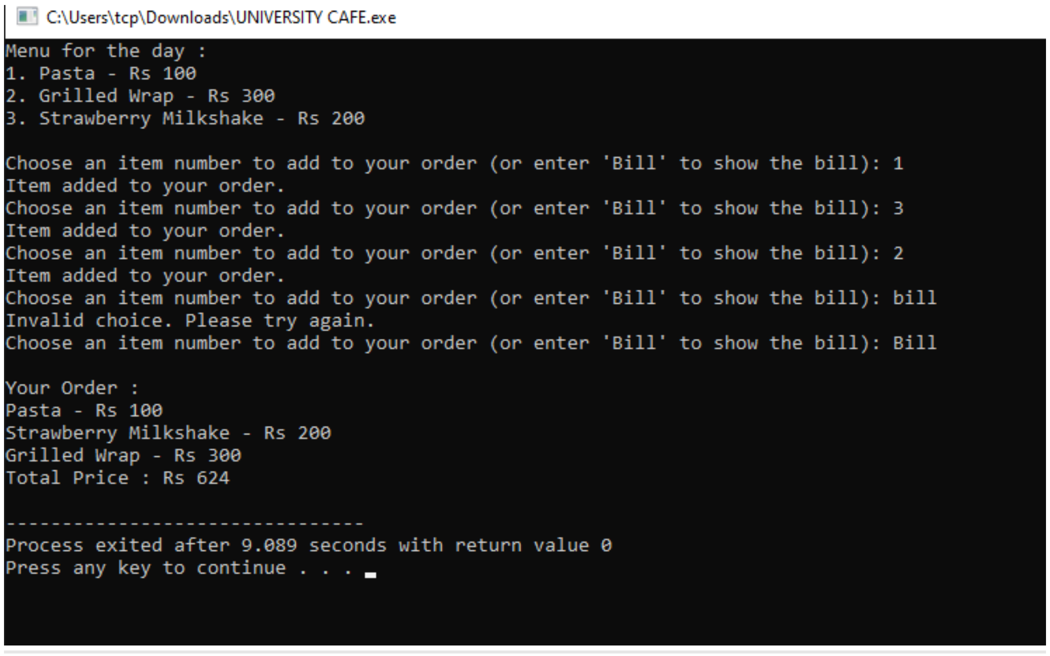
Transport output



C:\Users\tcp\Downloads\TRANSPORT No 2.exe

```
Route added successfully!
Route added successfully!
Route added successfully!
Transport System Menu:
1. Show Routes
2. Reserve a Seat
3. Show Last Reservation
4. Check Available Buses
5. Exit
Enter your choice: 2

Enter the Route ID: 1
Seat reserved successfully!
Route ID: 1
Departure: University
Arrival: Murree Road
Departure Time: 17:30
Bus Number: BUS001


Transport System Menu:
1. Show Routes
2. Reserve a Seat
3. Show Last Reservation
4. Check Available Buses
5. Exit
Enter your choice:
```

University café output



C:\Users\tcp\Downloads\UNIVERSITY CAFE.exe

```
Menu for the day :
1. Pasta - Rs 100
2. Grilled Wrap - Rs 300
3. Strawberry Milkshake - Rs 200

Choose an item number to add to your order (or enter 'Bill' to show the bill): 1
Item added to your order.
Choose an item number to add to your order (or enter 'Bill' to show the bill): 3
Item added to your order.
Choose an item number to add to your order (or enter 'Bill' to show the bill): 2
Item added to your order.
Choose an item number to add to your order (or enter 'Bill' to show the bill): bill
Invalid choice. Please try again.
Choose an item number to add to your order (or enter 'Bill' to show the bill): Bill

Your Order :
Pasta - Rs 100
Strawberry Milkshake - Rs 200
Grilled Wrap - Rs 300
Total Price : Rs 624

--------------------------------
Process exited after 9.089 seconds with return value 0
Press any key to continue . . .
```

LEARNING OUTCOMES

C++ program code (Include it after learning outcomes as an appendix, If the code is very

lengthy, upload it on GitHub and add the link of the uploaded file in the appendix).