**COMPILER CONSTRUCTION**

**MID LAB**

**Name :** Hussain Ali

**Reg No. :** FA21-BCS-066

**Section :** BCS-7A

**QUESTION NO. 1**

**Source Code:**

#include <iostream>

#include <string>

#include <vector>

#include <cstdlib>

using namespace std;

string registeredUsername = "movieLover";

string registeredPassword = "ticketSecure";

void displayMainMenu();

bool verifyUserCredentials();

void enableGuestMode();

void chooseMovie();

void presentMovieDetails(int movieID);

void bookSeat(const string& movieName, const string& showtime);

void handlePayment(const string& movieName, const string& showtime);

bool inquireAboutAnotherBooking();

void showSeatArrangement(const vector<vector<char>>& seatingArrangement);

int main() {

bool continueBooking = true;

while (continueBooking) {

displayMainMenu();

int selectedOption;

cin >> selectedOption;

switch (selectedOption) {

case 1:

if (verifyUserCredentials()) {

chooseMovie();

}

break;

case 2:

enableGuestMode();

chooseMovie();

break;

case 3:

cout << "Thank you for using the system. Goodbye!\n";

return 0;

default:

cout << "Invalid option, please select a valid choice.\n";

}

continueBooking = inquireAboutAnotherBooking();

}

return 0;

}

void displayMainMenu() {

cout << "==== Welcome to the Cinema Ticket Booking System ====\n";

cout << "1. Enter Username and Password\n";

cout << "2. Guest Checkout\n";

cout << "3. Exit\n";

cout << "Select an option: ";

}

bool verifyUserCredentials() {

string inputUsername, inputPassword;

int attemptsLeft = 3;

while (attemptsLeft > 0) {

cout << "Username: ";

cin >> inputUsername;

cout << "Password: ";

cin >> inputPassword;

if (inputUsername == registeredUsername && inputPassword == registeredPassword) {

cout << "Login Successful! Enjoy your movie experience.\n";

return true;

}

else {

attemptsLeft--;

cout << "Invalid login. Attempts remaining: " << attemptsLeft << "\n";

}

}

cout << "Login Failed. The program will exit now.\n";

return false;

}

void enableGuestMode() {

cout << "You are now in Guest Mode. Please note that features may be limited.\n";

}

void chooseMovie() {

int movieChoice;

cout << "Select a movie from the list below:\n";

cout << "1. Enigmatic Thriller\n";

cout << "2. Family Fun\n";

cout << "3. Futuristic Sci-Fi\n";

cout << "4. Exit\n";

cout << "Your choice: ";

cin >> movieChoice;

if (movieChoice == 4) {

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

return;

}

switch (movieChoice) {

case 1:

presentMovieDetails(movieChoice);

bookSeat("Enigmatic Thriller", "5:00 PM");

break;

case 2:

presentMovieDetails(movieChoice);

bookSeat("Family Fun", "6:30 PM");

break;

case 3:

presentMovieDetails(movieChoice);

bookSeat("Futuristic Sci-Fi", "8:00 PM");

break;

default:

cout << "Invalid movie selection. Please try again.\n";

}

}

void presentMovieDetails(int movieID) {

switch (movieID) {

case 1:

cout << "Movie: Enigmatic Thriller | Genre: Thriller | Duration: 140 mins | Showtimes: 5:00 PM\n";

break;

case 2:

cout << "Movie: Family Fun | Genre: Family | Duration: 100 mins | Showtimes: 6:30 PM\n";

break;

case 3:

cout << "Movie: Futuristic Sci-Fi | Genre: Sci-Fi | Duration: 150 mins | Showtimes: 8:00 PM\n";

break;

default:

cout << "Invalid movie selection. Please try again.\n";

}

}

void bookSeat(const string& movieName, const string& showtime) {

vector<vector<char>> seatingArrangement(5, vector<char>(6, 'O'));

int rowChoice, columnChoice;

while (true) {

showSeatArrangement(seatingArrangement);

cout << "Select your seat by entering row and column numbers (e.g., 0 2): ";

cin >> rowChoice >> columnChoice;

if (rowChoice < 0 || rowChoice >= seatingArrangement.size() || columnChoice < 0 || columnChoice >= seatingArrangement[0].size()) {

cout << "Invalid seat choice. Please try again.\n";

}

else if (seatingArrangement[rowChoice][columnChoice] == 'O') {

seatingArrangement[rowChoice][columnChoice] = 'X';

cout << "Seat successfully reserved!\n";

handlePayment(movieName, showtime);

break;

}

else {

cout << "This seat is already taken. Please select a different one.\n";

}

}

}

void showSeatArrangement(const vector<vector<char>>& seatingArrangement) {

cout << "Seat Map (O = Available, X = Occupied):\n";

for (const auto& row : seatingArrangement) {

for (const auto& seat : row) {

cout << seat << " ";

}

cout << endl;

}

}

void handlePayment(const string& movieName, const string& showtime) {

int paymentMethod;

cout << "Choose your payment option:\n";

cout << "1. Credit Card\n";

cout << "2. Mobile Wallet\n";

cout << "3. Cancel Booking\n";

cin >> paymentMethod;

switch (paymentMethod) {

case 1:

case 2:

cout << "Payment successful! Your booking is confirmed.\n";

cout << "Movie: " << movieName << "\n";

cout << "Showtime: " << showtime << "\n";

cout << "Reference ID: " << rand() % 10000 + 1000 << "\n";

break;

case 3:

cout << "Booking has been cancelled.\n";

break;

default:

cout << "Invalid option, please choose again.\n";

}

}

bool inquireAboutAnotherBooking() {

string reply;

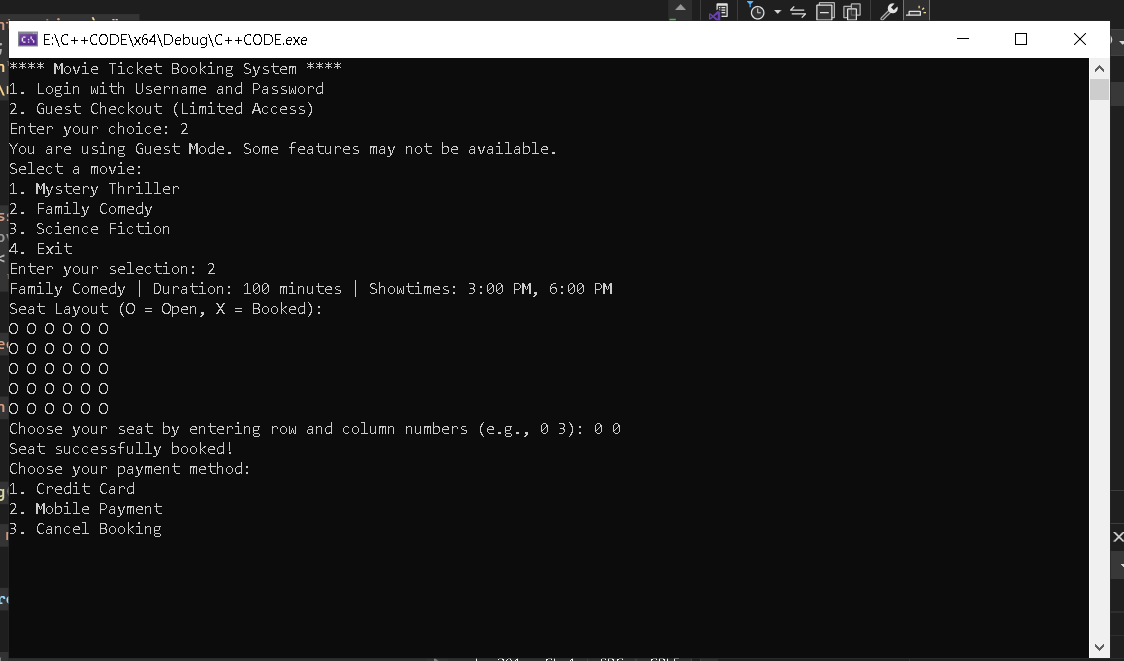
cout << "Would you like to make another booking? (Yes/No): ";

cin >> reply;

return (reply == "Yes" || reply == "yes");

}

**OUTPUT:**

****

**Question No. 2**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace WindowsFormsApp2

{

public partial class Form1 : Form

{

private static readonly HashSet<string> cppKeywords = new HashSet<string>

{

"int", "void", "bool", "string", "while", "if", "else", "return", "using", "namespace",

"cout", "cin", "include", "vector", "true", "false", "case", "break"

};

private static readonly HashSet<string> cppOperators = new HashSet<string>

{

"+", "-", "\*", "/", "=", "==", ">", "<", ">=", "<=", "&&", "||", "!", "++", "--"

};

private static readonly Regex varPattern = new Regex(@"^[a-zA-Z\_][a-zA-Z0-9\_]\*$");

private static readonly Regex opPattern = new Regex(@"[\+\-\\*\/\=\>\<\&\|\!]{1,2}");

private static readonly Regex strPattern = new Regex(@"^""[^""]\*""$");

public Form1()

{

InitializeComponent();

}

private void SubmitButton\_Click(object sender, EventArgs e)

{

TokenizedOutput.Clear();

string sourceCode = SourceCode.Text;

List<string> tokens = TokenizeSourceCode(sourceCode);

DisplayTokens(tokens);

}

private List<string> TokenizeSourceCode(string code)

{

List<string> extractedTokens = new List<string>();

string[] words = code.Split(new[] { ' ', '\n', '\t', ';', '(', ')', '{', '}', '[', ']', ',', '.' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var word in words)

{

if (cppKeywords.Contains(word))

extractedTokens.Add($"Keyword: {word}");

else if (cppOperators.Contains(word))

extractedTokens.Add($"Operator: {word}");

else if (varPattern.IsMatch(word))

extractedTokens.Add($"Variable: {word}");

else if (strPattern.IsMatch(word))

extractedTokens.Add($"String: {word}");

else if (opPattern.IsMatch(word))

extractedTokens.Add($"Operator: {word}");

else

extractedTokens.Add($"Identifier: {word}");

}

return extractedTokens;

}

private void DisplayTokens(List<string> tokens)

{

foreach (var token in tokens)

{

TokenizedOutput.AppendText(token + Environment.NewLine);

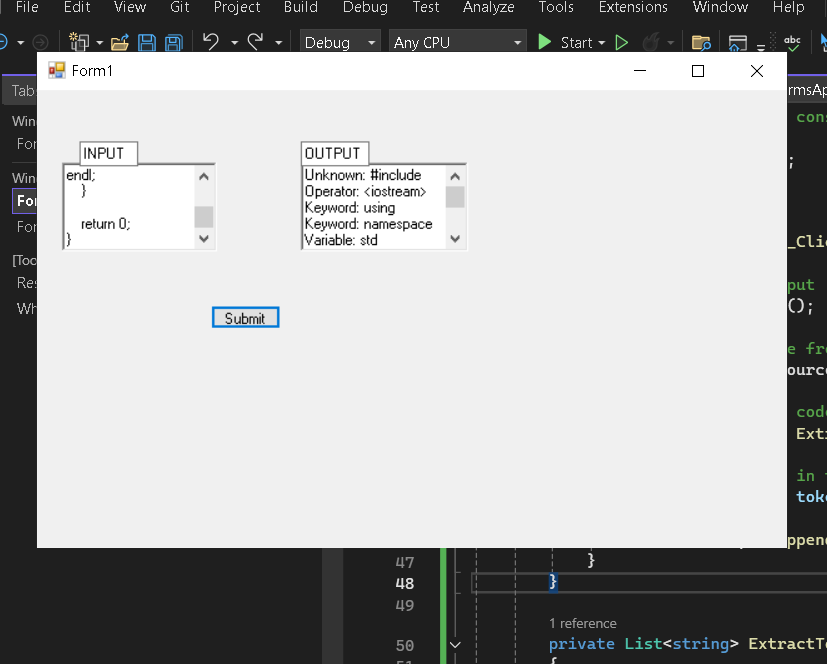
}

}

}

}

**Output:**

****

**-----------------------THE END----------------------**