Riddhi Jain Task-8

1) Login/Signup System using C++

A-

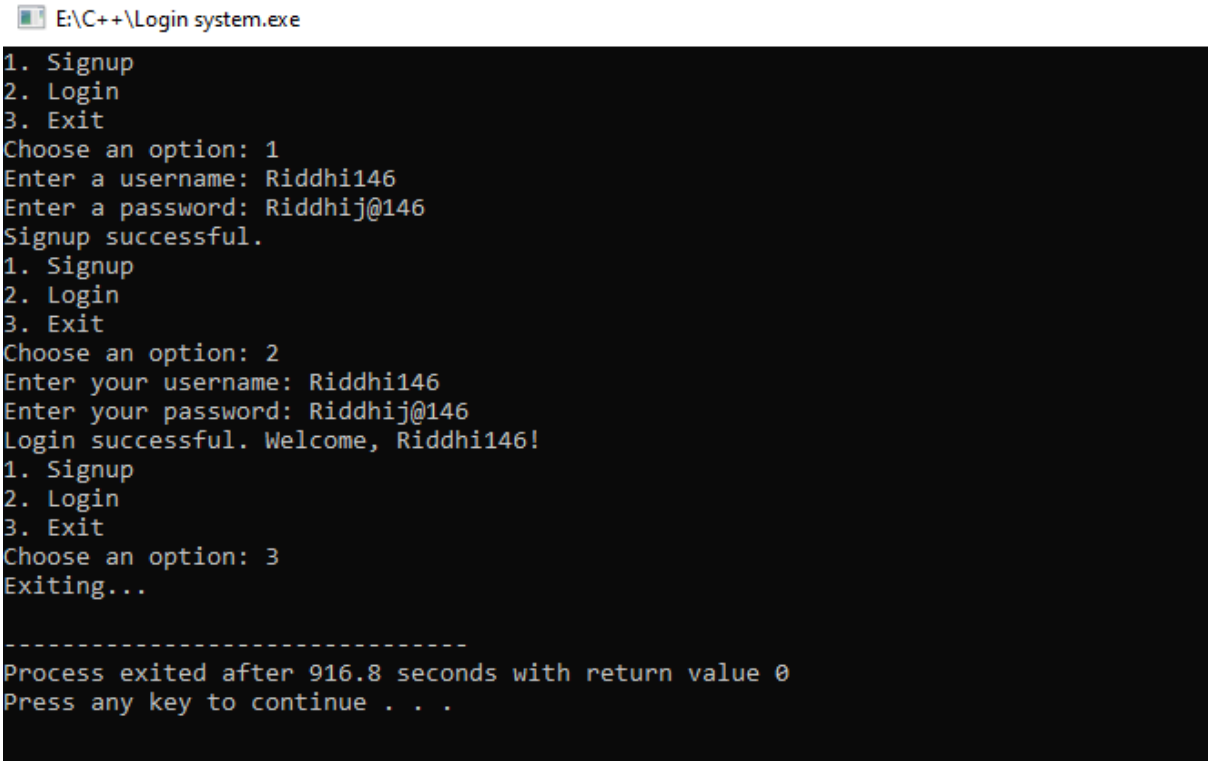
Login system.cpp

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
bool userExists(const string& username) {
    ifstream file("users.txt");
    string line;
    while (getline(file, line)) {
        size_t delimiter_pos = line.find(':');
        if (delimiter_pos != string::npos) {
            string stored_username = line.substr(0, delimiter_pos);
            if (stored_username == username) {
                return true;
            }
        }
    }
    return false;
}
bool authenticateUser(const string& username, const string& password) {
    ifstream file("users.txt");
    string line;
    while (getline(file, line)) {
        size_t delimiter_pos = line.find(':');
        if (delimiter_pos != string::npos) {
            string stored_username = line.substr(0, delimiter_pos);
            string stored_password = line.substr(delimiter_pos + 1);
            if (stored_username == username && stored_password == password) {
                return true;
            }
        }
    }
    return false;
}
```

```cpp
void signup() {
    string username, password;
    cout << "Enter a username: ";
    cin >> username;
    if (userExists(username)) {
        cout << "Username already exists. Please try again.\n";
        return;
    }
    cout << "Enter a password: ";
    cin >> password;

    ofstream file("users.txt", ios::app);
    file << username << ":" << password << "\n";
    file.close();
    cout << "Signup successful.\n";
}
void login() {
    string username, password;
    cout << "Enter your username: ";
    cin >> username;
    cout << "Enter your password: ";
    cin >> password;

    if (authenticateUser(username, password)) {
        cout << "Login successful. Welcome, " << username << "!\n";
    } else {
        cout << "Invalid username or password. Please try again.\n";
    }
}
```

```cpp
int main() {
    int choice;
    while (true) {
        cout << "1. Signup\n2. Login\n3. Exit\nChoose an option: ";
        cin >> choice;

        switch (choice) {
            case 1:
                signup();
                break;
            case 2:
                login();
                break;
            case 3:
                cout << "Exiting...\n";
                return 0;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    }
}

```

Output:-



2) Bookshop Inventory System using C++

A-

[*] Bookshop_inventory.cpp

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
using namespace std;
struct Book {
    string title;
    string author;
    string isbn;
    double price;
    int quantity;
};
vector<Book> loadInventory(const string& filename) {
    vector<Book> inventory;
    ifstream file(filename);
    string line;
    while (getline(file, line)) {
        istringstream iss(line);
        Book book;
        getline(iss, book.title, ',');
        getline(iss, book.author, ',');
        getline(iss, book.isbn, ',');
        iss >> book.price;
        iss.ignore(1, ',');
        iss >> book.quantity;
        inventory.push_back(book);
    }
    return inventory;
}
void saveInventory(const vector<Book>& inventory, const string& filename) {
    ofstream file(filename);
    for (const auto& book : inventory) {
        file << book.title << "," << book.author << "," << book.isbn << ","
             << book.price << "," << book.quantity << "\n";
    }
}
```

```cpp
void addBook(vector<Book>& inventory) {
    Book book;
    cout << "Enter title: ";
    cin.ignore();
    getline(cin, book.title);
    cout << "Enter author: ";
    getline(cin, book.author);
    cout << "Enter ISBN: ";
    cin >> book.isbn;
    cout << "Enter price: ";
    cin >> book.price;
    cout << "Enter quantity: ";
    cin >> book.quantity;
    inventory.push_back(book);
    cout << "Book added successfully.\n";
}
void removeBook(vector<Book>& inventory) {
    string isbn;
    cout << "Enter ISBN of the book to remove: ";
    cin >> isbn;
    auto it = remove_if(inventory.begin(), inventory.end(), [&](const Book& book) {
        return book.isbn == isbn;
    });
    if (it != inventory.end()) {
        inventory.erase(it, inventory.end());
        cout << "Book removed successfully.\n";
    } else {
        cout << "Book not found.\n";
    }
}
```

```cpp
void updateBook(vector<Book>& inventory) {
    string isbn;
    cout << "Enter ISBN of the book to update: ";
    cin >> isbn;
    for (auto& book : inventory) {
        if (book.isbn == isbn) {
            cout << "Enter new title (or press enter to keep unchanged): ";
            cin.ignore();
            string newTitle;
            getline(cin, newTitle);
            if (!newTitle.empty()) book.title = newTitle;
            cout << "Enter new author (or press enter to keep unchanged): ";
            string newAuthor;
            getline(cin, newAuthor);
            if (!newAuthor.empty()) book.author = newAuthor;
            cout << "Enter new price (or press enter to keep unchanged): ";
            string newPrice;
            getline(cin, newPrice);
            if (!newPrice.empty()) book.price = stod(newPrice);
            cout << "Enter new quantity (or press enter to keep unchanged): ";
            string newQuantity;
            getline(cin, newQuantity);
            if (!newQuantity.empty()) book.quantity = stoi(newQuantity);
            cout << "Book updated successfully.\n";
            return;
        }
    }
    cout << "Book not found.\n";
}
```

```cpp
void displayInventory(const vector<Book>& inventory) {
    if (inventory.empty()) {
        cout << "No books in inventory.\n";
        return;
    }
    for (const auto& book : inventory) {
        cout << "Title: " << book.title << "\n"
             << "Author: " << book.author << "\n"
             << "ISBN: " << book.isbn << "\n"
             << "Price: $" << book.price << "\n"
             << "Quantity: " << book.quantity << "\n"
             << "----------------------\n";
    }
}
```

```cpp
int main() {
    vector<Book> inventory = loadInventory("inventory.txt");
    int choice;
    while (true) {
        cout << "1. Add Book\n2. Remove Book\n3. Update Book\n4. Display Inventory\n5. Exit\nChoose an option: ";
        cin >> choice;

        switch (choice) {
            case 1:
                addBook(inventory);
                break;
            case 2:
                removeBook(inventory);
                break;
            case 3:
                updateBook(inventory);
                break;
            case 4:
                displayInventory(inventory);
                break;
            case 5:
                saveInventory(inventory, "inventory.txt");
                cout << "Exiting...\n";
                return 0;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    }
}
```

3) Sudoku Game using C++

A-

Sudoku.cpp

```cpp
#include <iostream>
#include <vector>
#define N 9
class Sudoku {
private:
    int board[N][N];
    bool isValid(int row, int col, int num) {
        for (int x = 0; x < N; x++) {
            if (board[row][x] == num || board[x][col] == num || board[row - row % 3 + x / 3][col - col % 3 + x % 3] == num) {
                return false;
            }
        }
        return true;
    }
    bool isSolved() {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (board[i][j] == 0) {
                    return false;
                }
            }
        }
        return true;
    }
public:
    Sudoku() {
        int initialBoard[N][N] = {
            {5, 3, 0, 0, 7, 0, 0, 0, 0},
            {6, 0, 0, 1, 9, 5, 0, 0, 0},
            {0, 9, 8, 0, 0, 0, 0, 6, 0},
            {8, 0, 0, 0, 6, 0, 0, 0, 3},
            {4, 0, 0, 8, 0, 3, 0, 0, 1},
            {7, 0, 0, 0, 2, 0, 0, 0, 6},
            {0, 6, 0, 0, 0, 0, 2, 8, 0},
            {0, 0, 0, 4, 1, 9, 0, 0, 5},
            {0, 0, 0, 0, 8, 0, 0, 7, 9}
        };
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                board[i][j] = initialBoard[i][j];
            }
        }
    }
    void displayBoard() {
        for (int i = 0; i < N; i++) {
            if (i % 3 == 0 && i != 0) {
                std::cout << "----------------------\n";
            }
            for (int j = 0; j < N; j++) {
                if (j % 3 == 0 && j != 0) {
                    std::cout << " | ";
                }
                std::cout << board[i][j] << " ";
            }
            std::cout << "\n";
        }
    }
```

```
    bool makeMove(int row, int col, int num) {
        if (board[row][col] == 0 && isValid(row, col, num)) {
            board[row][col] = num;
            return true;
        }
        return false;
    }

    bool gameWon() {
        return isSolved();
    }
};
int main() {
    Sudoku game;
    int row, col, num;
    while (!game.gameWon()) {
        game.displayBoard();
        std::cout << "Enter your move (row col num): ";
        std::cin >> row >> col >> num;

        if (row < 0 || col < 0 || num < 0) {
            std::cout << "Exiting game...\n";
            break;
        }

        if (game.makeMove(row, col, num)) {
            std::cout << "Move accepted.\n";
        } else {
            std::cout << "Invalid move. Try again.\n";
        }
    }
    if (game.gameWon()) {
        std::cout << "Congratulations! You've solved the Sudoku puzzle.\n";
    }
    return 0;
}
```

Output:-

```
E:\C++\Sudoku.exe
5 3 0  | 0 7 0  | 0 0 0
6 0 0  | 1 9 5  | 0 0 0
0 9 8  | 0 0 0  | 0 6 0
---------------------
8 0 0  | 0 6 0  | 0 0 3
4 0 0  | 8 0 3  | 0 0 1
7 0 0  | 0 2 0  | 0 0 6
---------------------
0 6 0  | 0 0 0  | 2 8 0
0 0 0  | 4 1 9  | 0 0 5
0 0 0  | 0 8 0  | 0 7 9
Enter your move (row col num): 3
5
8
Invalid move. Try again.
5 3 0  | 0 7 0  | 0 0 0
6 0 0  | 1 9 5  | 0 0 0
0 9 8  | 0 0 0  | 0 6 0
---------------------
8 0 0  | 0 6 0  | 0 0 3
4 0 0  | 8 0 3  | 0 0 1
7 0 0  | 0 2 0  | 0 0 6
---------------------
0 6 0  | 0 0 0  | 2 8 0
0 0 0  | 4 1 9  | 0 0 5
0 0 0  | 0 8 0  | 0 7 9
Enter your move (row col num):
```