```
#include <iostream>
#include <fstream>
#include <string>
#include <stack>
#include <cstdlib>
using namespace std;


#define RESET   "\033[0m"
#define CYAN    "\033[36m"
#define RED     "\033[31m"
#define BLUE    "\033[34m"
#define YELLOW  "\033[33m"
#define GREEN   "\033[32m"

const int MAX_USERS = 100;
const int MAX_INTERESTS = 10;


class InterestNode {
public:
    string interest;
    InterestNode* left;
    InterestNode* right;

    InterestNode(string val) : interest(val), left(nullptr), right(nullptr) {}
};

//Interest tree
class InterestTree {
public:
    InterestNode* root;

    InterestTree() : root(nullptr) {}

    int countRec(InterestNode* node) {
    if (!node) return 0;
    return 1 + countRec(node->left) + countRec(node->right);
}

int count() {
    return countRec(root);
}
```

```
// Recursion of interes input
InterestNode* insertRec(InterestNode* node, const string& interest) {
    if (!node) return new InterestNode(interest);
    if (interest < node->interest)
        node->left = insertRec(node->left, interest);
    else
        node->right = insertRec(node->right, interest);
    return node;
}


void insert(const string& interest) {
    root = insertRec(root, interest);
}


void displayRec(InterestNode* node) {
    if (!node) return;
    displayRec(node->left);
    cout << "- " << node->interest << endl;
    displayRec(node->right);
}

//to display interests
void display() {
    if (!root) {
        cout << RED << "\t\t\tNo interests added yet." << RESET << endl;
    } else {
        cout << YELLOW << "\t\t\tInterests:" << RESET << endl;
        displayRec(root);
    }
}

// Find common interests between two interest trees
bool hasInterest(InterestNode* node, const string& interest) {
    if (!node) return false;
    if (node->interest == interest) return true;
    if (interest < node->interest)
        return hasInterest(node->left, interest);
    else
        return hasInterest(node->right, interest);
}

bool contains(const string& interest) {
```

```
        return hasInterest(root, interest);
    }
};


class User;

//friendship linked list
struct FriendEdge {
    User* friendUser;
    FriendEdge* next;

    FriendEdge(User* user) : friendUser(user), next(nullptr) {}
};


class User {
public:
    string username;
    FriendEdge* friendList;
    stack<string> posts;
    InterestTree interests;

    User() : username(""), friendList(nullptr) {}
    User(string uname) : username(uname), friendList(nullptr) {}

    //Linked list to adding friend
    void addFriend(User* other) {

        FriendEdge* current = friendList;
        while (current) {
            if (current->friendUser == other) {
                return; // Friend already exists
            }
            current = current->next;
        }

        FriendEdge* newEdge = new FriendEdge(other);
        newEdge->next = friendList;
        friendList = newEdge;
    }

    // Display all friends
    void displayFriends() {
```

```
    if (!friendList) {
        cout << RED << "\t\t\t " << username << " has no friends yet." << RESET << endl;
        return;
    }
    cout << YELLOW << "\t\t\t " << username << "'s Friends:" << RESET << endl;
    FriendEdge* temp = friendList;
    while (temp) {
        cout << "- " << temp->friendUser->username << endl;
        temp = temp->next;
    }
}

// Create a new post
void createPost(const string& content) {
    posts.push(content);
}

// Display posts from top of stack(most recent posts would be posted on top)
void displayPosts() {
    if (posts.empty()) {
        cout << RED << "\t\t\t " << username << " has no posts yet." << RESET << endl;
        return;
    }
    cout << YELLOW << "\t\t\t " << username << "'s Posts:" << RESET << endl;
    stack<string> temp = posts;
    while (!temp.empty()) {
        cout << CYAN << "\t\t\t> " << temp.top() << RESET << endl;
        temp.pop();
    }
}

// Add a new interest
void addInterest(const string& interest) {
    interests.insert(interest);
}

// Display interests
void showInterests() {
    interests.display();
}

// Check for common interests with another user
bool hasCommonInterests(User* otherUser) {
    if (!otherUser) return false;
```

```
        // Check each interest in this user's tree against the other user's tree
        stack<InterestNode*> nodeStack;
        InterestNode* current = interests.root;

        while (current || !nodeStack.empty()) {
            while (current) {
                nodeStack.push(current);
                current = current->left;
            }

            current = nodeStack.top();
            nodeStack.pop();

            if (otherUser->interests.contains(current->interest)) {
                return true;
            }

            current = current->right;
        }

        return false;
    }
};


class SocialMediaApp {
public:
    User users[MAX_USERS];
    int userCount;

    SocialMediaApp() : userCount(0) {}

    void run();

    // Add a new user
    void addUser(const string& username) {
        // Check if user already exists
        for (int i = 0; i < userCount; i++) {
            if (users[i].username == username) {
                return; // User already exists
            }
        }
```

```
        if (userCount >= MAX_USERS) {
            cout << RED << "\t\t\tUser limit reached!" << RESET << endl;
            return;
        }
        users[userCount++] = User(username);
    }

    // Get user by username
    User* getUser(const string& username) {
        for (int i = 0; i < userCount; i++) {
            if (users[i].username == username)
                return &users[i];
        }
        return nullptr;
    }

    User* findUser(const string& username) {
        return getUser(username);
    }

    // Menu functions
    void findUserMenu();
    void connectFriends();
    void viewFriendList();
    void createPostMenu();
    void viewPostsMenu();
    void showInterestsMenu();
    void addInterestToUser();
    void initializeFriendConnections();
};

// So it wont accept numbers or special characters
bool isValidInterest(const string& interest) {
    for (char c : interest) {

        if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || c == ' ')) {
            return false;
        }
    }
    return true;
}

// Main interactive menu
void SocialMediaApp::run() {
```

```
    int choice;
    do {
        system("cls");
        cout << GREEN << "\t\t\t\t\t================== LinkLoop Menu
==================\n";
        cout << "\t\t\t\t\t\t\t1. Find Users\n";
        cout << "\t\t\t\t\t\t\t2. Add Friends\n";
        cout << "\t\t\t\t\t\t\t3. View Friend List\n";
        cout << "\t\t\t\t\t\t\t4. Create Post\n";
        cout << "\t\t\t\t\t\t\t5. View Posts\n";
        cout << "\t\t\t\t\t\t\t6. Show Interests\n";
        cout << "\t\t\t\t\t\t\t7. Add interests to user\n";
        cout << "\t\t\t\t\t\t\t8. Exit\n";
        cout << GREEN <<
"\t\t\t\t\t==================================================\n";
        cout << BLUE << "\t\t\tEnter choice: " << RESET;
        cin >> choice;

        switch (choice) {
            case 1:
                findUserMenu();
                break;
            case 2:
                connectFriends();
                break;
            case 3:
                viewFriendList();
                break;
            case 4:
                createPostMenu();
                break;
            case 5:
                viewPostsMenu();
                break;
            case 6:
                showInterestsMenu();
                break;
            case 7:
                addInterestToUser();
                break;
            case 8:
                cout << RED << "\t\t\tExiting to login menu..." << RESET << endl;
                break;
            default:
```

```
            cout << RED << "\t\t\tInvalid choice. Try again." << RESET << endl;
            break;
        }
        if (choice != 8) system("pause");
    } while (choice != 8);
}

// To find another user
void SocialMediaApp::findUserMenu() {
    cout << YELLOW << "\t\t\tPeople you may know:" << RESET << endl;
    bool found = false;

    // List all users in the list
    for (int i = 0; i < userCount; i++) {
        User* user = &users[i];
        cout << "\t\t\t- " << user->username << endl;
        found = true;
    }

    if (!found) {
        cout << RED << "\t\t\tNo users found." << RESET << endl;
        return;
    }

    // Ask the user to select one of the users shown
    string selectedUsername;
    cout << CYAN << "\t\t\tEnter the username to view details: " << RESET;
    cin >> selectedUsername;

    User* selectedUser = findUser(selectedUsername);
    if (selectedUser) {
        cout << GREEN << "\t\t\tShowing details for " << selectedUsername << ":" << RESET <<
endl;

        // Display user's interests
        cout << YELLOW << "\t\t\t " << selectedUsername << "'s Interests:" << RESET << endl;
        selectedUser->showInterests();

        // Display posts
        selectedUser->displayPosts();

        // Display friends
        selectedUser->displayFriends();
```

```
        // Show friends with common interests
        cout << YELLOW << "\t\t\tFriends with common interests:" << RESET << endl;
        FriendEdge* friendEdge = selectedUser->friendList;
        bool foundCommonInterests = false;

        while (friendEdge) {
            User* friendUser = friendEdge->friendUser;
            if (selectedUser->hasCommonInterests(friendUser)) {
                cout << "\t\t\t- " << friendUser->username << endl;
                foundCommonInterests = true;
            }
            friendEdge = friendEdge->next;
        }

        if (!foundCommonInterests) {
            cout << RED << "\t\t\tNo friends with common interests found." << RESET << endl;
        }
    } else {
        cout << RED << "\t\t\tUser not found." << RESET << endl;
    }
}


// Connects two users as friends
void SocialMediaApp::connectFriends() {
    cout << YELLOW << "\t\t\tUsers available to connect with:" << RESET << endl;
    bool found = false;

    // List all users in the system
    for (int i = 0; i < userCount; i++) {
        User* user = &users[i];
        cout << "\t\t\t- " << user->username << endl;
        found = true;
    }

    if (!found) {
        cout << RED << "\t\t\tNo users found." << RESET << endl;
        return;
    }

    // Ask the user to select a friend to connect with
    string user1, user2;
    cout << CYAN << "\t\t\tEnter your username: " << RESET;
    cin >> user1;
```

```
    cout << CYAN << "\t\t\tEnter the username of the user you want to connect with: " << RESET;
    cin >> user2;

    User* u1 = findUser(user1);
    User* u2 = findUser(user2);

    if (u1 && u2 && u1 != u2) {
        u1->addFriend(u2);
        u2->addFriend(u1);
        cout << GREEN << "\t\t\tYou are now friends and connected!" << RESET << endl;
    } else if (u1 == u2) {
        cout << RED << "\t\t\tYou cannot connect with yourself." << RESET << endl;
    } else {
        cout << RED << "\t\t\tOne or both users not found." << RESET << endl;
    }
}


// Shows user's friends
void SocialMediaApp::viewFriendList() {
    string username;
    cout << CYAN << "\t\t\tEnter username to view friends: " << RESET;
    cin >> username;
    User* user = findUser(username);
    if (user)
        user->displayFriends();
    else
        cout << RED << "\t\t\tUser not found." << RESET << endl;
}

// Lets user create a new post
void SocialMediaApp::createPostMenu() {
    string username, content;
    cout << CYAN << "\t\t\tEnter your username: " << RESET;
    cin >> username;
    User* user = findUser(username);
    if (user) {
        cout << CYAN << "\t\t\tEnter post content: " << RESET;
        cin.ignore();
        getline(cin, content);
        user->createPost(content);
        cout << GREEN << "\t\t\tPost created!" << RESET << endl;
    } else {
        cout << RED << "\t\t\tUser not found." << RESET << endl;
```

```
    }
}

// Displays posts of a user
void SocialMediaApp::viewPostsMenu() {
    string username;
    cout << CYAN << "\t\t\tEnter username to view posts: " << RESET;
    cin >> username;
    User* user = findUser(username);
    if (user)
        user->displayPosts();
    else
        cout << RED << "\t\t\tUser not found." << RESET << endl;
}

// To add more interests to the people in the system
void SocialMediaApp::addInterestToUser() {
    cout << YELLOW << "\t\t\tPeople You Can Add:" << RESET << endl;
    for (int i = 0; i < userCount; ++i) {
        cout << "\t\t\t- " << users[i].username << endl;
    }

    string username;
    cout << CYAN << "\t\t\tEnter your username to add interests: " << RESET;
    cin >> username;
    cin.ignore();

    User* user = findUser(username);
    if (!user) {
        cout << RED << "\t\t\tUser not found." << RESET << endl;
        return;
    }

    int currentInterestCount = user->interests.count();
    int remaining = 10 - currentInterestCount;

    if (remaining <= 0) {
        cout << RED << "\t\t\tYou have already reached the 10-interest limit." << RESET << endl;
        return;
    }

    cout << CYAN << "\t\t\tYou can add up to " << remaining << " more interests." << RESET <<
endl;
```

```
    for (int i = 0; i < remaining; ++i) {
        string interest;
        cout << CYAN << "\t\t\tEnter interest (or type 'done' to stop): " << RESET;
        getline(cin, interest);
        if (interest == "done") break;

        // Validate the interest
        if (!isValidInterest(interest)) {
            cout << RED << "\t\t\tInvalid interest! Please use only letters and spaces." << RESET <<
endl;
            i--; // Decrement counter to retry this slot
            continue;
        }

        user->addInterest(interest);
        cout << GREEN << "\t\t\tInterest \"" << interest << "\" added." << RESET << endl;
    }
}

// Displays user's interests
void SocialMediaApp::showInterestsMenu() {
    string username;
    cout << CYAN << "\t\t\tEnter username to show interests: " << RESET;
    cin >> username;
    User* user = findUser(username);
    if (user)
        user->showInterests();
    else
        cout << RED << "\t\t\tUser not found." << RESET << endl;
}

//Login/Registration Functions

// Function for logging in
void login(SocialMediaApp& app) {
    string username, password, un, pass;
    int count = 0;
    system("cls");

    cout << "\t\t\t Enter USERNAME: ";
    cin >> username;
    cout << "\t\t\t Enter PASSWORD: ";
    cin >> password;
```

```
    ifstream input("info.txt");
    while (input >> un >> pass) {
        if (un == username && pass == password) {
            count = 1;
            break;
        }
    }
    input.close();

    if (count == 1) {
        cout << "\t\t\t\nLOGIN SUCCESSFUL!\n";
        if (!app.getUser(username)) {
            app.addUser(username);
        }
        app.run();
    } else {
        cout << "\t\t\t\nINVALID USERNAME OR PASSWORD\n";
        system("pause");
    }
}

//Registration with interest validation and multi-word support
void registration(SocialMediaApp& app) {
    string username, password;
    cout << "\t\t\tEnter username: ";
    cin >> username;
    cout << "\t\t\tEnter password: ";
    cin >> password;
    cin.ignore(); // Clear the newline character

    ofstream output("info.txt", ios::app);
    output << username << " " << password << endl;
    output.close();
    cout << GREEN << "\t\t\tRegistration successful!\n";

    app.addUser(username);

    string interest;
    int entered = 0;
    cout << CYAN << "\t\t\tEnter up to 10 interests (type 'done' to finish early):" << RESET <<
endl;
    while (entered < MAX_INTERESTS) {
        cout << "\t\t\t- Interest: ";
        getline(cin, interest);
```

```
        if (interest == "done" || interest == "DONE") break;

        // Validate the interest
        if (!isValidInterest(interest)) {
            cout << RED << "\t\t\tInvalid interest! Please use only letters and spaces." << RESET <<
endl;
            continue; // Skip this input and ask again
        }

        app.getUser(username)->addInterest(interest);
        entered++;
    }
    if (entered == MAX_INTERESTS) {
        cout << RED << "\t\t\tYou have reached the maximum number of interests allowed.\n" <<
RESET;
    }
    system("pause");
}


// Forgot password functionality
void forgot() {
    string searchUsername, searchPassword, un, pass;
    int found = 0;
    system("cls");

    cout << "\t\t\t\nEnter the username you remember: ";
    cin >> searchUsername;

    ifstream input("info.txt");
    while (input >> un >> pass) {
        if (un == searchUsername) {
            found = 1;
            break;
        }
    }
    input.close();

    if (found) {
        cout << "\t\t\t\nAccount found!\n";
        cout << "\t\t\tYour password is: " << pass << endl;
    } else {
        cout << "\t\t\t\nSorry, account not found.\n";
    }
```

```
    system("pause");
}
//incremented users and interests in the app
void SocialMediaApp::initializeFriendConnections() {

    getUser("Jenny")->addFriend(getUser("Hana"));
    getUser("Hana")->addFriend(getUser("Jenny"));

    getUser("Hana")->addFriend(getUser("Vince"));
    getUser("Vince")->addFriend(getUser("Hana"));

    getUser("Hana")->addFriend(getUser("Khe"));
    getUser("Khe")->addFriend(getUser("Hana"));

    getUser("Charlie")->addFriend(getUser("Leanne"));
    getUser("Leanne")->addFriend(getUser("Charlie"));

    getUser("Charlie")->addFriend(getUser("Vince"));
    getUser("Vince")->addFriend(getUser("Charlie"));

    getUser("Charlie")->addFriend(getUser("Khe"));
    getUser("Khe")->addFriend(getUser("Charlie"));

    getUser("Charlie")->addFriend(getUser("Gorge"));
    getUser("Gorge")->addFriend(getUser("Charlie"));

    getUser("Jazmin")->addFriend(getUser("Dianne"));
    getUser("Dianne")->addFriend(getUser("Jazmin"));

    getUser("Jazmin")->addFriend(getUser("Eraiza"));
    getUser("Eraiza")->addFriend(getUser("Jazmin"));

    getUser("Jazmin")->addFriend(getUser("Hana"));
    getUser("Hana")->addFriend(getUser("Jazmin"));

    getUser("Jazmin")->addFriend(getUser("Jenny"));
    getUser("Jenny")->addFriend(getUser("Jazmin"));
}

//main function
int main() {
    SocialMediaApp app;

    string usernames[] = {
```

```
        "Hana", "Vince", "Charlie", "Dianne",
        "Jazmin", "Gorge", "Eraiza", "Khe", "Jenny", "Leanne"
    };
    string interests[][MAX_INTERESTS] = {
        {"music", "reading", "hiking", "coding"},
        {"gaming", "movies", "sports", "technology", "dog"},
        {"cooking", "yoga", "traveling", "photography"},
        {"painting", "poetry", "dancing", "singing", "cat"},
        {"robotics", "novels", "chess", "designs"},
        {"fashion", "makeup", "styling", "shopping", "astrology"},
        {"fitness", "running", "cycling", "nutrition"},
        {"programming", "dog", "literature", "math"},
        {"anime", "comics", "cosplay", "manga", "manhwa", "digital art"},
        {"gardening", "pets", "DIY", "nature","foraging"}
    };

    for (int i = 0; i < 10; ++i) {
        app.addUser(usernames[i]);
        User* user = app.getUser(usernames[i]);
        for (int j = 0; j < MAX_INTERESTS; ++j) {
            if (j < MAX_INTERESTS && !interests[i][j].empty()) {
                user->addInterest(interests[i][j]);
            }
        }
    }

    // Connection of friends
    app.initializeFriendConnections();

    // Main menu loop for login system
    int choice;
    while(true) {
        system("cls");
        cout << "\t\t\t\t\t================== Welcome to LinkLoop ==================\n";
        cout << "\t\t\t\t\t\t\t1. Login\n";
        cout << "\t\t\t\t\t\t\t2. Register\n";
        cout << "\t\t\t\t\t\t\t3. Forgot Password\n";
        cout << "\t\t\t\t\t\t\t4. Exit\n";
        cout <<
"\t\t\t\t\t============================================================\n";
        cout << "\t\t\t\t\t\t\tEnter your choice: ";
        cin >> choice;

        switch (choice) {
```

```
            case 1:
                login(app);
                break;
            case 2:
                registration(app);
                break;
            case 3:
                forgot();
                break;
            case 4:
                cout << "\t\t\tExiting..." << endl;
                return 0;
            default:
                cout << "\t\t\tInvalid choice. Try again.\n";
                system("pause");
                break;
        }
    }

    return 0;
}
```