Question 2

Social Network Management System:

Data Structures: Graph (adjacency list) for user connections, linked list for user profiles.

Functionality:

Create user profiles (name, interests, etc.).

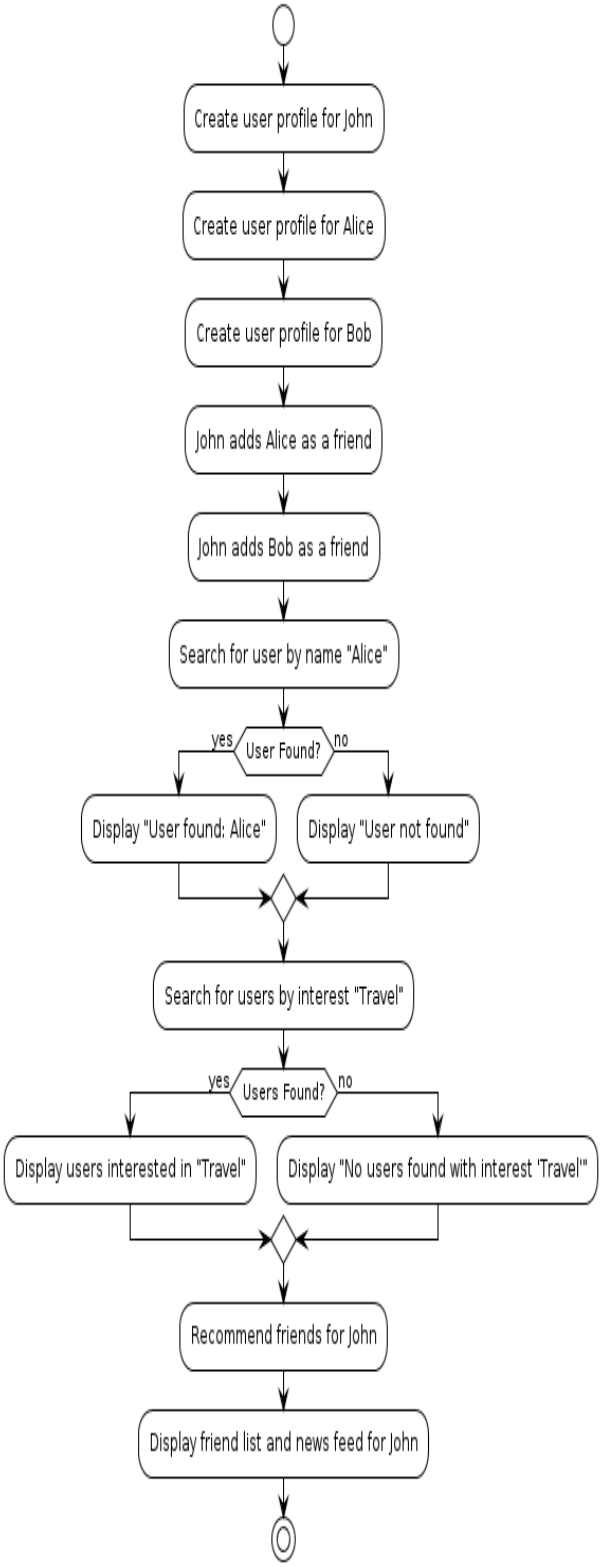
Add friends (connect users in the graph).

Search for users by name or interest.

Recommend friends based on mutual connections or interests.

Display a user's friend list and news feed (simulated data or integration with an external API).

User Profile Creation, Connection Establishment, Search, Recommendation, and Display



Network.c

```
#include <stdio.h>
#include <string.h>
#include "social_network.h"

void add_user(Graph *graph, const char *name, const char *interests) {
    if (graph->num_users >= MAX_USERS) {
        printf("Cannot add more users.\n");
        return;
    }
    User *user = &graph->users[graph->num_users++];
    strcpy(user->name, name);
    strcpy(user->interests, interests);
    user->num_friends = 0;
}

void add_friendship(Graph *graph, const char *user1_name, const char *user2_name) {
    User *user1 = NULL, *user2 = NULL;
    for (int i = 0; i < graph->num_users; i++) {
        if (strcmp(graph->users[i].name, user1_name) == 0) {
            user1 = &graph->users[i];
        }
        if (strcmp(graph->users[i].name, user2_name) == 0) {
            user2 = &graph->users[i];
        }
    }
    if (user1 && user2) {
        user1->friends[user1->num_friends++] = user2;
        user2->friends[user2->num_friends++] = user1;
    }
}
```

```
void search_users_by_name(Graph *graph, const char *name) {
    printf("Users with name '%s':\n", name);
    for (int i = 0; i < graph->num_users; i++) {
        if (strstr(graph->users[i].name, name)) {
            printf("%s\n", graph->users[i].name);
        }
    }
}

void search_users_by_interest(Graph *graph, const char *interest) {
    printf("Users interested in '%s':\n", interest);
    for (int i = 0; i < graph->num_users; i++) {
        if (strstr(graph->users[i].interests, interest)) {
            printf("%s\n", graph->users[i].name);
        }
    }
}

void recommend_friends(Graph *graph, const char *name) {
    printf("Recommended friends for %s:\n", name);
    for (int i = 0; i < graph->num_users; i++) {
        if (strcmp(graph->users[i].name, name) == 0) {
            User *user = &graph->users[i];
            for (int j = 0; j < user->num_friends; j++) {
                User *friend = user->friends[j];
                for (int k = 0; k < friend->num_friends; k++) {
                    User *potential_friend = friend->friends[k];
                    if (strcmp(potential_friend->name, name) != 0 &&
                        !is_friend_of_user(user, potential_friend)) {
                        printf("%s\n", potential_friend->name);
                    }
                }
            }
            break;
        }
```

```
    }
}

bool is_friend_of_user(User *user, User *potential_friend) {
    for (int i = 0; i < user->num_friends; i++) {
        if (user->friends[i] == potential_friend) {
            return true;
        }
    }
    return false;
}
```

Network.h

```
#ifndef SOCIAL_NETWORK_H
#define SOCIAL_NETWORK_H

#define MAX_USERS 100
#define MAX_NAME_LENGTH 50
#define MAX_INTEREST_LENGTH 100

typedef struct User {
    char name[MAX_NAME_LENGTH];
    char interests[MAX_INTEREST_LENGTH];
    struct User *friends[MAX_USERS];
    int num_friends;
} User;

typedef struct Graph {
    User users[MAX_USERS];
    int num_users;
} Graph;

void add_user(Graph *graph, const char *name, const char *interests);
void add_friendship(Graph *graph, const char *user1_name, const char *user2_name);
void search_users_by_name(Graph *graph, const char *name);
```

```
void search_users_by_interest(Graph *graph, const char *interest);
void recommend_friends(Graph *graph, const char *name);
void display_user_friends(User *user);
void display_news_feed(User *user);

#endif
```

Networkmain.c

```
[17:12, 03/05/2024] Nandu: #include <stdio.h>
#include <string.h>
#include "social_network.h"

void add_user(Graph *graph, const char *name, const char *interests) {
    if (graph->num_users >= MAX_USERS) {
        printf("Cannot add more users.\n");
        return;
    }
    User *user = &graph->users[graph->num_users++];
    strcpy(user->name, name);
    strcpy(user->interests, interests);
    user->num_friends = 0;
}

void add_friendship(Graph *graph, const char *user1_name, const char *user2_name) {
    User *user1 = NULL, *user2 = NULL;
    for (int i = 0; i < graph->num_users; i++) {
        if (strcmp(graph->users[i].name, user1_name) == 0) {
            user1 = &graph->users[i];
        }
        if (strcmp(graph->users[i].name, user2_name) == 0) {
            user2 = &graph->users[i];
        }
    }
```

```
    if (user1 && user2) {
        user1->friends[user1->num_friends++] = user2;
        user2->friends[user2->num_friends++] = user1;
    }
}

void search_users_by_name(Graph *graph, const char *name) {
    printf("Users with name '%s':\n", name);
    for (int i = 0; i < graph->num_users; i++) {
        if (strstr(graph->users[i].name, name)) {
            printf("%s\n", graph->users[i].name);
        }
    }
}

void search_users_by_interest(Graph *graph, const char *interest) {
    printf("Users interested in '%s':\n", interest);
    for (int i = 0; i < graph->num_users; i++) {
        if (strstr(graph->users[i].interests, interest)) {
            printf("%s\n", graph->users[i].name);
        }
    }
}

void recommend_friends(Graph *graph, const char *name) {
    printf("Recommended friends for %s:\n", name);
    for (int i = 0; i < graph->num_users; i++) {
        if (strcmp(graph->users[i].name, name) == 0) {
            User *user = &graph->users[i];
            for (int j = 0; j < user->num_friends; j++) {
                User *friend = user->friends[j];
                for (int k = 0; k < friend->num_friends; k++) {
                    User *potential_friend = friend->friends[k];
                    if (strcmp(potential_friend->name, name) != 0 &&
                        !is_friend_of_user(user, potential_friend)) {
```

```
                printf("%s\n", potential_friend->name);
            }
        }
    }
    break;
  }
 }
}

bool is_friend_of_user(User *user, User *potential_friend) {
  for (int i = 0; i < user->num_friends; i++) {
    if (user->friends[i] == potential_friend) {
      return true;
    }
  }
  return false;
}
[17:12, 03/05/2024] Nandu: .c
[17:12, 03/05/2024] Nandu: #include <stdio.h>
#include <stdlib.h>
#include "social_network.h"
#include <time.h>

int main() {
  Graph social_network;
  social_network.num_users = 0;

  // Create users
  add_user(&social_network, "Alice", "coding, reading");
  add_user(&social_network, "Bob", "sports, cooking");
  add_user(&social_network, "Charlie", "coding, gaming");

  // Add friendships
  add_friendship(&social_network, "Alice", "Bob");
  add_friendship(&social_network, "Bob", "Charlie");
```

```
    // Search for users
    search_users_by_name(&social_network, "Alice");
    search_users_by_interest(&social_network, "coding");

    // Recommend friends
    recommend_friends(&social_network, "Alice");

    return 0;
}
```

Explanation:

User Profiles:

Each user profile can be represented by a node in a graph.

Additionally, user profile information such as name, interests, etc., can be stored in a linked list or a similar data structure attached to each node.

Graph for User Connections:

The user connections can be represented using a graph data structure with an adjacency list implementation.

Each user is a node in the graph, and edges represent connections (friendships) between users.

The adjacency list for each user node contains references to other users to whom they are connected.

Creating User Profiles:

When creating a user profile, you would add a new node to the graph to represent the user.

The user's profile information can be stored in a linked list or similar data structure attached to the user node.

Adding Friends:

Adding a friend connection between two users involves adding an edge between their respective nodes in the graph.

Searching for Users:

Searching for users by name or interest involves traversing the graph and looking for nodes that match the search criteria.

This can be done using various graph traversal algorithms like depth-first search (DFS) or breadth-first search (BFS).

Recommend Friends:

Friend recommendations can be based on mutual connections or shared interests.

To recommend friends based on mutual connections, you can look for users who are connected to multiple mutual friends.

To recommend friends based on shared interests, you can look for users who have similar interests to the target user.

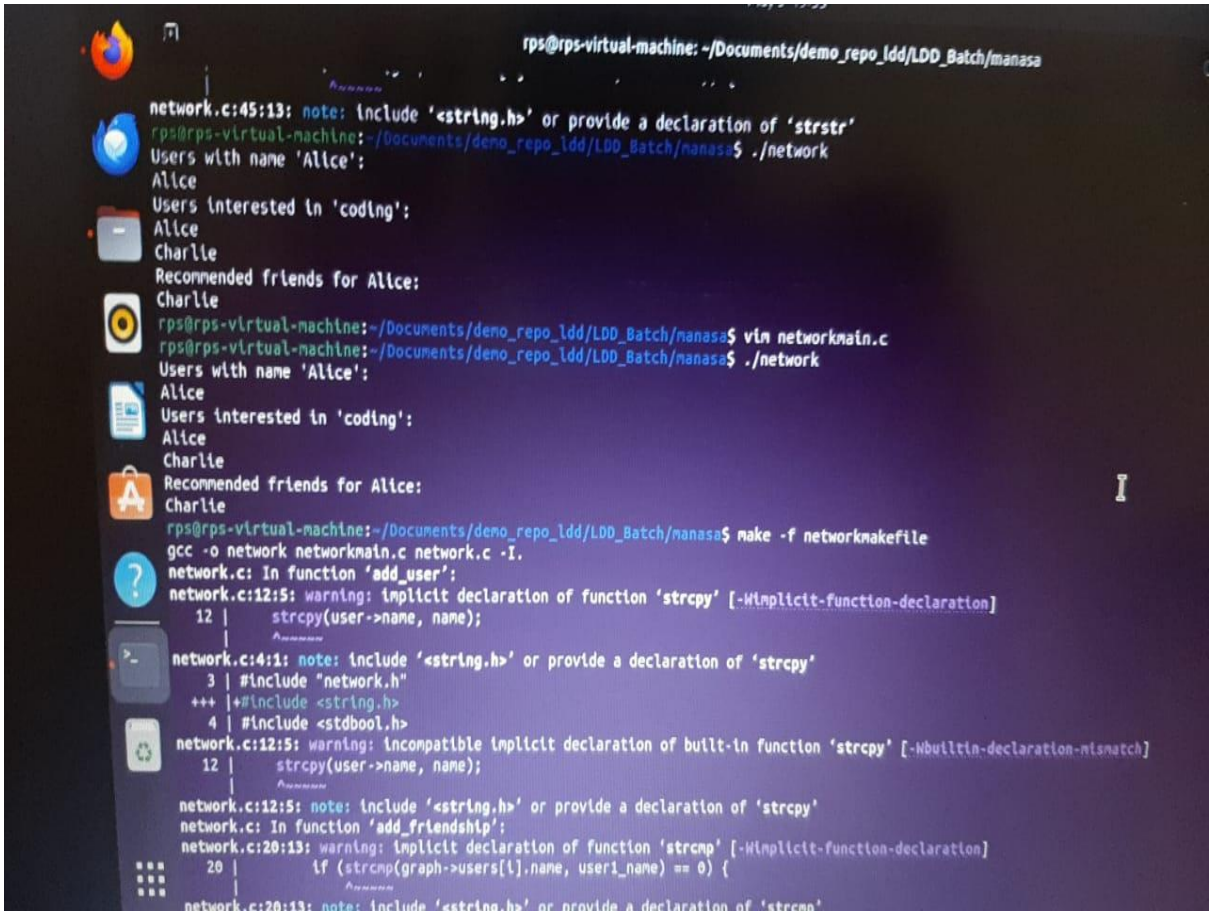
Displaying Friend List and News Feed:

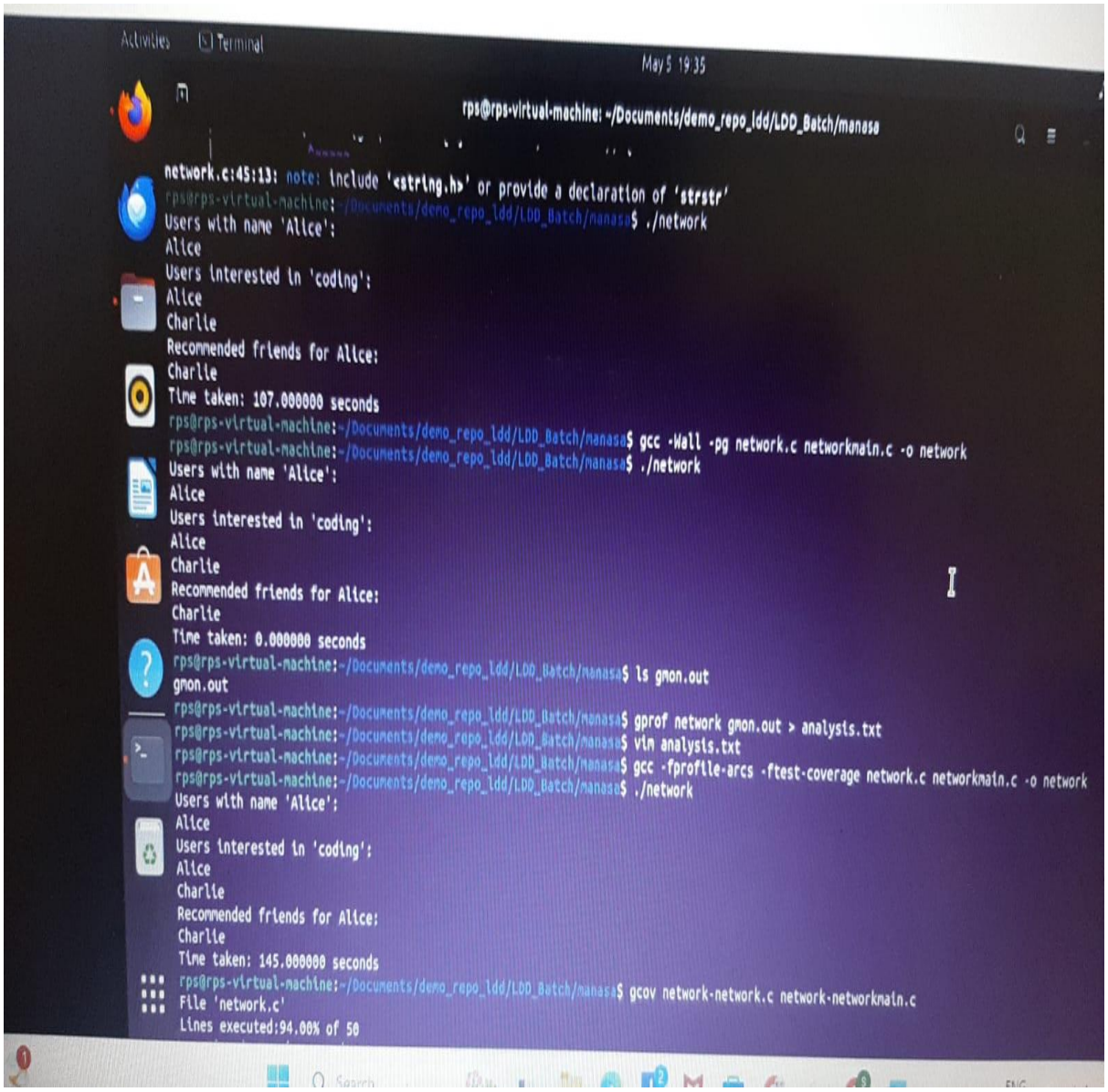
A user's friend list can be obtained by traversing the adjacency list of their node in the graph.

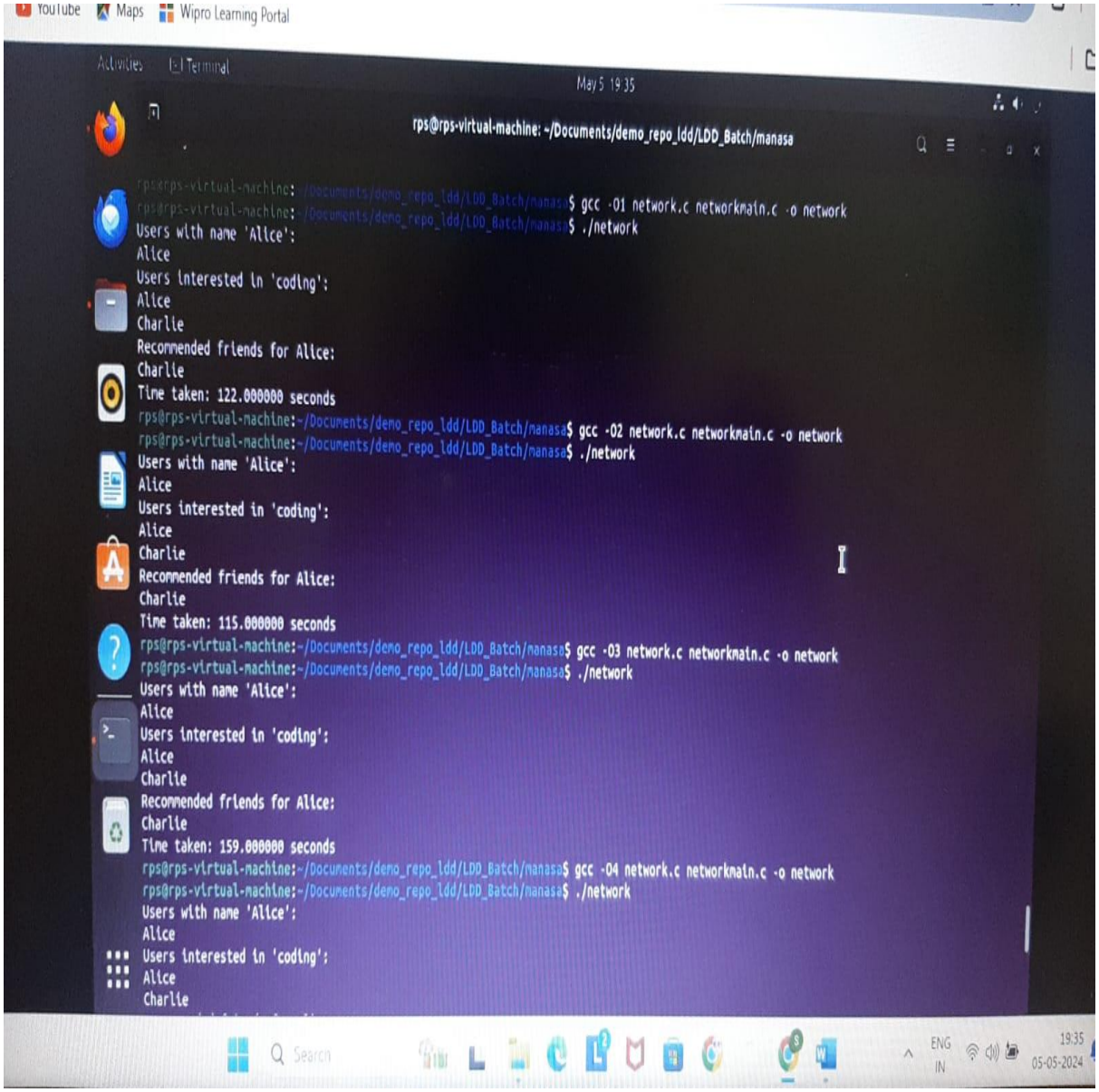
The news feed can be simulated data generated within the system or integrated with an external API to fetch real-time updates from friends.

By implementing these functionalities, you can create a Social Network Management System that allows users to connect with each other, search for other users, receive friend recommendations, and interact with their friends' content.

output


rps@rps-virtual-machine: ~/Documents/demo_repo_ldd/LDD_Batch/manasa
network.c:45:13: note: include '<string.h>' or provide a declaration of 'strstr'
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ vim networkmain.c
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ make -f networkmakefile
gcc -o network networkmain.c network.c -I.
network.c: In function 'add_user':
network.c:12:5: warning: implicit declaration of function 'strcpy' [-Wimplicit-function-declaration]
12 | strcpy(user->name, name);
network.c:4:1: note: include '<string.h>' or provide a declaration of 'strcpy'
3 | #include "network.h"
+++ |+#include <string.h>
4 | #include <stdbool.h>
network.c:12:5: warning: incompatible implicit declaration of built-in function 'strcpy' [-Wbuiltin-declaration-mismatch]
12 | strcpy(user->name, name);
network.c:12:5: note: include '<string.h>' or provide a declaration of 'strcpy'
network.c: In function 'add_friendship':
network.c:20:13: warning: implicit declaration of function 'strcmp' [-Wimplicit-function-declaration]
20 | if (strcmp(graph->users[i].name, user1_name) == 0) {


Activities
Terminal
May 5 19:35
rps@rps-virtual-machine: ~/Documents/demo_repo_ldd/LDD_Batch/manasa
network.c:45:13: note: include '<string.h>' or provide a declaration of 'strstr'
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 107.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -Wall -pg network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 0.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ls gmon.out
gmon.out
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gprof network gmon.out > analysis.txt
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ vim analysis.txt
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -fprofile-arcs -ftest-coverage network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 145.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcov network-network.c network-networkmain.c
File 'network.c'
Lines executed:94.00% of 50


YouTube
Maps
Wipro Learning Portal
Activities
Terminal
May 5 19:35
rps@rps-virtual-machine: ~/Documents/demo_repo_ldd/LDD_Batch/manasa
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -O1 network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 122.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -O2 network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 115.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -O3 network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Recommended friends for Alice:
Charlie
Time taken: 159.000000 seconds
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ gcc -O4 network.c networkmain.c -o network
rps@rps-virtual-machine:~/Documents/demo_repo_ldd/LDD_Batch/manasa$ ./network
Users with name 'Alice':
Alice
Users interested in 'coding':
Alice
Charlie
Search
ENG
IN
19:35
05-05-2024