**BHAVANA**

**03/05/2024**

**PROJECT 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).

**Graph.h**

#ifndef GRAPH\_H //Header guard to prevent multiple inclusion

#define GRAPH\_H //preprocessor directive the deines macro GRAPH\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//define constants for maximum user and friends

#define MAX\_USERS 100

#define MAX\_FRIENDS 100

//define the structure for a user profile

typedef struct {

char name[50];//name of the user

char interests[100];//Interests of the user

} UserProfile;

//define the strucure for a friend node

typedef struct FriendNode {

int userId;//Id of the friend

struct FriendNode\* next;//pointer to the next friend in the list

} FriendNode;

//Define the structure for the social network

typedef struct {

int totalUsers;//total number of users in the network

UserProfile profiles[MAX\_USERS];//Array to store user profiles

FriendNode\* friends[MAX\_USERS];//Array to store friend lists for each user

} SocialNetwork;

//Function prototypes

FriendNode\* createFriendNode(int userId);//Function to create a new friend node

SocialNetwork\* createSocialNetwork();//function to create a new social network

void addUserProfile(SocialNetwork\* network, char\* name, char\* interests);//Function to adddd a new user profile

void addFriend(SocialNetwork\* network, char\* userName1, char\* userName2);//Function to add a friend connection

int findUserByName(SocialNetwork\* network, char\* name);//function to find a user by name

void recommendFriends(SocialNetwork\* network, char\* userName);//function to recommend friends for a user

void displayFriendList(SocialNetwork\* network, char\* userName);//function to display the friend list for a user

void displayNewsFeed();//function to display the news feed

#endif /\* GRAPH\_H\*///end of the header guard

**EXPLANATION:**

Sure, here's a brief explanation of the provided code:

1. \*\*Header Guard\*\*: The `#ifndef`, `#define`, and `#endif` lines ensure that the content of the header file is included only once in each compilation unit, preventing multiple inclusions that could lead to compilation errors.

2. \*\*Includes\*\*: The necessary standard library headers `<stdio.h>`, `<stdlib.h>`, and `<string.h>` are included for standard I/O, memory allocation, and string manipulation functions, respectively.

3. \*\*Constants\*\*: Constants `MAX\_USERS` and `MAX\_FRIENDS` are defined to specify the maximum number of users and friends allowed in the social network.

4. \*\*Structures\*\*:

- `UserProfile`: Represents a user profile with a name and interests.

- `FriendNode`: Represents a node in a linked list of friends, containing the ID of the friend and a pointer to the next friend.

- `SocialNetwork`: Contains the total number of users, an array of user profiles, and an array of friend lists for each user.

5. \*\*Function Prototypes\*\*: Prototypes for various functions are declared, including functions for creating friend nodes, creating a social network, adding user profiles, adding friend connections, finding users by name, recommending friends, displaying friend lists, and displaying a news feed.

6. \*\*Function Definitions\*\*: The function definitions are not included in the header file but should be implemented in a corresponding source file (`graph.c` or similar). These functions would provide the actual functionality for managing the social network, such as creating nodes, adding users, adding friends, etc.

7. \*\*End of Header Guard\*\*: Ensures the end of the header guard section.

Overall, this header file provides a framework for implementing a simple social network system in C, including user profiles, friend connections, and basic functionalities like adding users, adding friends, and recommending friends.

**Graph.c:**

#include "graph.h" // Including the header file for the Social Network Management System

FriendNode\* createFriendNode(int userId) { // Function to create a new friend node for a given user ID

FriendNode\* node = (FriendNode\*)malloc(sizeof(FriendNode)); // Allocating memory for the new friend node

if (node == NULL) { // Checking if memory allocation failed

printf("Memory allocation failed.\n"); // Printing an error message

exit(EXIT\_FAILURE); // Exiting the program with an error code

}

node->userId = userId; // Setting the user ID for the new friend node

node->next = NULL; // Initializing the next pointer to NULL

return node; // Returning the newly created friend node

}

SocialNetwork\* createSocialNetwork() { // Function to create a new social network

SocialNetwork\* network = (SocialNetwork\*)malloc(sizeof(SocialNetwork)); // Allocating memory for the new social network

if (network == NULL) { // Checking if memory allocation failed

printf("Memory allocation failed.\n"); // Printing an error message

exit(EXIT\_FAILURE); // Exiting the program with an error code

}

network->totalUsers = 0; // Initializing the total number of users to 0

return network; // Returning the newly created social network

}

void addUserProfile(SocialNetwork\* network, char\* name, char\* interests) { // Function to add a new user profile to the social network

if (network->totalUsers >= MAX\_USERS) { // Checking if the maximum user limit has been reached

printf("Max user limit reached.\n"); // Printing a message indicating the limit has been reached

return; // Returning without adding the user profile

}

strcpy(network->profiles[network->totalUsers].name, name); // Copying the name to the new user profile

strcpy(network->profiles[network->totalUsers].interests, interests); // Copying the interests to the new user profile

network->friends[network->totalUsers] = NULL; // Initializing the friends list for the new user profile to NULL

network->totalUsers++; // Incrementing the total number of users in the social network

}

int findUserByName(SocialNetwork\* network, char\* name) { // Function to find a user in the social network by name

for (int i = 0; i < network->totalUsers; i++) { // Looping through all user profiles in the network

if (strcmp(network->profiles[i].name, name) == 0) { // Checking if the name matches

return i; // Returning the index of the user profile

}

}

return -1; // Returning -1 if the user was not found

}

void addFriend(SocialNetwork\* network, char\* userName1, char\* userName2) { // Function to add a friend connection between two users

int user1Index = findUserByName(network, userName1); // Finding the index of the first user

int user2Index = findUserByName(network, userName2); // Finding the index of the second user

if (user1Index == -1 || user2Index == -1) { // Checking if either user was not found

printf("User not found.\n"); // Printing an error message

return; // Returning without adding the friend connection

}

FriendNode\* friend1 = createFriendNode(user2Index); // Creating a new friend node for the second user

friend1->next = network->friends[user1Index]; // Adding the new friend node to the list of friends for the first user

network->friends[user1Index] = friend1; // Updating the list of friends for the first user

FriendNode\* friend2 = createFriendNode(user1Index); // Creating a new friend node for the first user

friend2->next = network->friends[user2Index]; // Adding the new friend node to the list of friends for the second user

network->friends[user2Index] = friend2; // Updating the list of friends for the second user

}

void recommendFriends(SocialNetwork\* network, char\* userName) { // Function to recommend friends for a user

int userIndex = findUserByName(network, userName); // Finding the index of the user

if (userIndex == -1) { // Checking if the user was not found

printf("User not found.\n"); // Printing an error message

return; // Returning without recommending friends

}

for (int i = 0; i < network->totalUsers; i++) { // Looping through all user profiles in the network

if (i != userIndex) { // Checking if the user is not the same as the current user

FriendNode\* current = network->friends[userIndex]; // Getting the list of friends for the current user

int isFriend = 0; // Flag to indicate if the current user is already a friend

while (current != NULL) { // Looping through the list of friends

if (current->userId == i) { // Checking if the current user is already a friend

isFriend = 1; // Setting the flag to true

break; // Exiting the loop

}

current = current->next; // Moving to the next friend in the list

}

if (!isFriend) { // Checking if the current user is not already a friend

printf("Recommendation: %s\n", network->profiles[i].name); // Printing the recommendation

}

}

}

}

void displayFriendList(SocialNetwork\* network, char\* userName) { // Function to display the friend list for a user

int userIndex = findUserByName(network, userName); // Finding the index of the user

if (userIndex == -1) { // Checking if the user was not found

printf("User not found.\n"); // Printing an error message

return; // Returning without displaying the friend list

}

FriendNode\* current = network->friends[userIndex]; // Getting the list of friends for the user

printf("Friend List of %s:\n", userName); // Printing the heading for the friend list

while (current != NULL) { // Looping through the list of friends

printf("- %s\n", network->profiles[current->userId].name); // Printing the name of each friend

current = current->next; // Moving to the next friend in the list

}

}

void displayNewsFeed() { // Function to display the news feed (placeholder)

// Placeholder for displaying news feed

printf("Displaying news feed...\n");

}

**EXPLANATION:**

This code provides the implementation of functions declared in the `graph.h` header file for managing a social network. Here's a brief overview of each function:

1. `createFriendNode`: Allocates memory for a new friend node, initializes it with the given user ID, and returns a pointer to it.

2. `createSocialNetwork`: Allocates memory for a new social network, initializes the total number of users to 0, and returns a pointer to it.

3. `addUserProfile`: Adds a new user profile to the social network with the given name and interests. Checks if the maximum user limit has been reached before adding.

4. `findUserByName`: Searches for a user in the social network by name and returns their index if found, otherwise returns -1.

5. `addFriend`: Adds a friend connection between two users specified by their names. It finds the indices of both users, creates friend nodes for each user, and adds them to each other's friend lists.

6. `recommendFriends`: Recommends friends for a user specified by their name. It checks all other users in the network and recommends those who are not already friends with the user.

7. `displayFriendList`: Displays the friend list for a user specified by their name. It finds the index of the user, traverses their friend list, and prints the names of their friends.

8. `displayNewsFeed`: Placeholder function for displaying the news feed. Currently, it only prints a placeholder message.

These functions provide basic functionality for managing user profiles, friend connections, and interactions within the social network. They can be used as building blocks for developing more complex features and interactions in a social networking application.

**Main.c:**

#include "graph.h"//Including the header file for the graph.h

#include <time.h>

int main() {

clock\_t start,end;

double cpu\_time\_used;

SocialNetwork\* network = createSocialNetwork();//creating a new social network

//Measure the time taken by the addUserProfile function

start=clock();

// Adding user profiles

addUserProfile(network, "Seetha", "Art, Music");//Adding user profile for seetha with interests

addUserProfile(network, "Ram", "Sports, Movies");//Adding user profile for ram with interests

addUserProfile(network, "Charlie", "Technology, Reading");//Adding user profile for Charlie with interests

addUserProfile(network, "Radha", "Gaming, Cooking");//Adding user profile for Radha with interests

addUserProfile(network, "Raman", "Travel, Photography");//Adding user profile foe Raman with interests

// Adding friendships

addFriend(network, "Seetha", "Ram");//seetha become friends with ram

addFriend(network, "Seetha", "Charlie");//seetha become friends with charlie

addFriend(network, "Seetha", "Radha");//seetha become friends with radha

addFriend(network, "Seetha", "Raman");//seetha become friends with raman

addFriend(network, "Ram", "Charlie");//ram become friends with charlie

addFriend(network, "Ram", "Radha");//ram become friends with radha

addFriend(network, "Ram", "Raman");//ram become friends with raman

addFriend(network, "Charlie", "Radha");//charlie become friends with radha

addFriend(network, "Charlie", "Raman");//charlie become friends with raman

//addFriend(network, "Radha", "Raman");//radha become friends with raman

// Displaying friend list

displayFriendList(network, "Seetha");//displaying seetha's friend list

displayFriendList(network, "Ram");//displaying ram's friend list

displayFriendList(network, "Charlie");//displaying charlie friend list

displayFriendList(network, "Radha");//displaying radha freind list

displayFriendList(network, "Raman");//displaying raman friend list

// Recommending friends

recommendFriends(network, "Seetha");//recommending friends for seetha

recommendFriends(network, "Ram");//recommending friends for ram

recommendFriends(network, "Charlie");//recommending freinds for charlie

recommendFriends(network, "Radha");//recommending friends for radha

recommendFriends(network, "Raman");//recommending friends for raman

// More functionality testing...

end=clock();

cpu\_time\_used=((double)(end-start))/CLOCKS\_PER\_SEC;

printf("Total time taken %f seconds to excute.\n",cpu\_time\_used);

return 0;//Returning 0 to indicate successful execution.

}

**EXPLANATION:**

Your `main` function demonstrates the usage of the functions defined in your `graph.h` header file to create a social network, add user profiles, establish friendships between users, display friend lists, recommend friends, and measure the execution time of adding user profiles. Here's a breakdown of what your `main` function does:

1. It includes necessary headers (`graph.h` and `time.h`).

2. It defines a `main` function.

3. It creates a new social network using the `createSocialNetwork` function.

4. It adds user profiles for "Seetha," "Ram," "Charlie," "Radha," and "Raman" along with their interests using the `addUserProfile` function.

5. It establishes friendships between users using the `addFriend` function. For example, "Seetha" becomes friends with "Ram," "Charlie," "Radha," and "Raman," and so on.

6. It displays the friend lists of each user using the `displayFriendList` function.

7. It recommends friends for each user using the `recommendFriends` function.

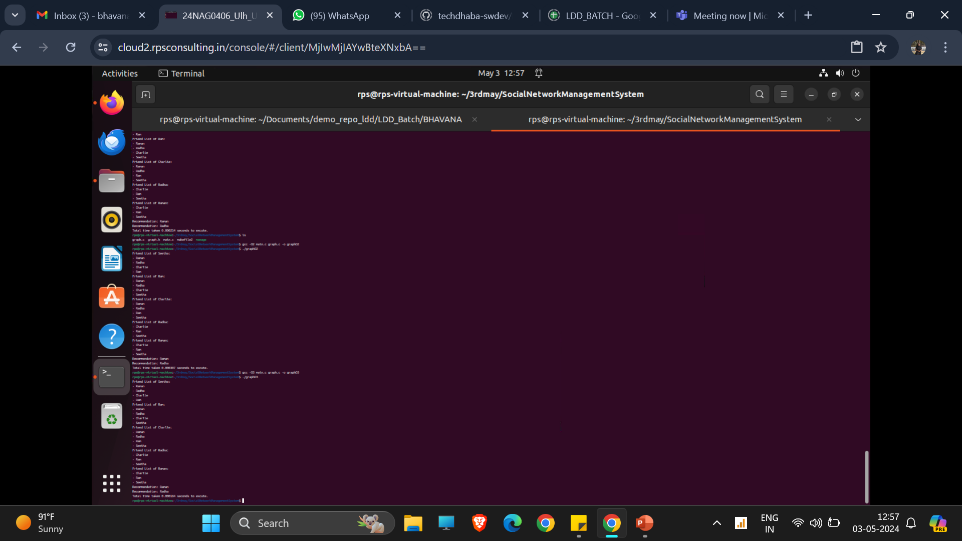
8. It measures the execution time taken by the `addUserProfile` function using the `clock` function and prints the total time taken in seconds.

9. Finally, it returns 0 to indicate successful execution.

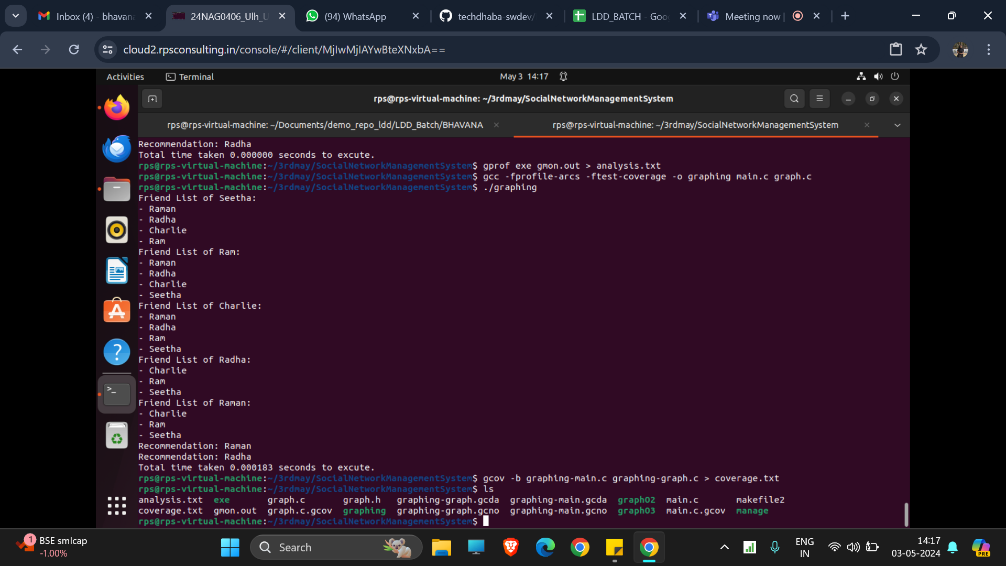
`main` function effectively tests the functionality of your social network management system by creating a network, adding users, establishing friendships, and performing other operations. The time measurement adds an extra dimension by assessing the performance of adding user profiles.

**OUTPUT:**

**OPTIMIZING THE CODE USING COMMAND:**

****

**CREATING ANALYSIS AND COVERAGE REPORT:**

****