Long Assignment 2: `SocialNet` Simulator

1 Introduction

The objective of this assignment is to build `SocialNet`, a command-line simulator for a social network's backend services. You will implement core functionalities for managing users, friendships, and content. The primary goal is to apply your understanding of AVL Trees and Graphs to a complex, practical application.

2 System Architecture

Your `SocialNet` will simulate a network of users, with each user being represented as a vertex in an undirected graph. The system must utilize the following core data structures:

- **Graphs:** To simulate a barebones social network, with each vertex representing a user, and an edge representing the friendship relation between the users corresponding to the neighbouring vertices.
- **AVL Tree:** To efficiently store the posts created by a user sorted according to the time of post creation.

Note: You must implement the above data structures (along with the operations needed for this project) yourself from scratch, i.e., you are **not** allowed to use C++ Libraries which already implement these data structures. However, you are free to use the C++ Library for Hashmaps / Dictionaries to map the username strings to the corresponding vertices in the graph.

3 Command Reference

Your program must read and execute a series of commands from `stdin`.

3.1 Social Network Operations

`ADD_USER <username>` Adds a new user to the network, initially with no friends and no posts.

`ADD_FRIEND <username1> <username2>` Establishes a bidirectional friendship between two existing usernames.

`LIST_FRIENDS <username>` Prints an alphabetically sorted list of the specified username's friends.

`SUGGEST_FRIENDS <username> <N>` Recommends up to `N` new friends who are "friends of a friend" but not already friends.

- **Ranking:** Recommendations are ranked by the number of mutual friends (descending). Ties are broken by alphabetical order of usernames.
- **Edge Cases:** If `N` is 0, output nothing. If fewer than `N` candidates exist, list all available.

`DEGREES_OF_SEPARATION <username1> <username2>` Calculates the length of the shortest path of friendships between two usernames. If no path exists, reports -1.

3.2 User Content Operations

ADD_POST <username> "<post_content>" Add a post whose content is the `post_content` string, to the posts created by the specified user.

OUTPUT_POSTS <username> <N> Output the most recent `N` posts of the user, in reverse chronological order. If `N` is -1, you should output all the posts by the user. If there are fewer than `N` posts by the user, then list all available posts.

Note: Usernames and post contents are **not** case-sensitive. For example, the usernames `Lakshay_col106` and `lakshay_COL106` are identical.

4 Submission

This is an individual assignment. You must submit a compressed file (.zip/.rar) containing your project code (.cpp, .hpp files, if any) along with a working shell script to compile your code. You must also add a README containing the instructions on how to run your code and use different commands. **Note** that the user must be able to input commands at runtime, i.e., from `stdin` and the commands must follow the given syntax.

The deadline for the submission is November 05, 15:00 IST (Wednesday). The submission link will be announced later.

5 Evaluation

The evaluation for the project will be based on a Viva which will involve (but not limited to) questions regarding your code, checking output on some specific sequence of commands, quality of the code etc.