By GALI RAVI PRAVEEN

PROBLEM STATEMENT

Alice is a programmer developing a social network website. The website has thousands of users. Alice must use a Data Structure to efficiently represent and store the users’ profiles. She needs to keep the information about how these users are connected with each other (connectivity means direct friendship in social media). Alice needs to find out the friends for a given user. This search needs to be executed quickly as there are thousands of users. She also needs to find the common friends for two given users. Propose a suitable Data Structure; implement the insert, delete, search operations and relevant algorithms to solve the problem. Analyse the time and space complexities of all the operations.

SOLUTION APPROACH

Step 1:

Created a structure which has a capacity to store the profile of the user.

i.e. User name, Age, place, Number of friends he/she has and for reference a ref\_no.

* Time Complexity to insert details is O (1).

Step 2:

The user structure is linked with the other users for traversing, using a linked list approach.

* Time complexity to insert the struct of user to a linked list is O (1).
* Time complexity to traverse to particular user is O(n).
* Space complexity is Θ(n) for average and for worst is O(n).

Step 3:

Using a linked list which is dynamic, we can create any number of users and also if users wanted to delete their account, we can easily do that.

* Deleting a linked list time complexity is O(n).

Step 4:

Can also get particular User info by giving that particular user’s reference number.

This can be achieved by using linked list traversing technique.

* Time complexity to traverse to particular user is O(n).

Step 5:

There is an option that to make friendship bonds between two friends, which takes input of that two friends reference number.

This was made by using GRAPHS which allows us to add edges between two selected users. AS the friendship is a bond on both sides, I used undirected graph here.

* Time complexity to add an edge is O (1).
* Space complexity is O(V+E).

Step 6:

Can also see all the present users of this application.

* Time complexity to add an edge is O (n).

Step 7:

In main function finally I wrote a switch case in a while loop which loops until user says quit.

Options are

1.Present users

2.Add user

3.User info

4.User friends

5.Make friends for USER

6.DELETE A USER

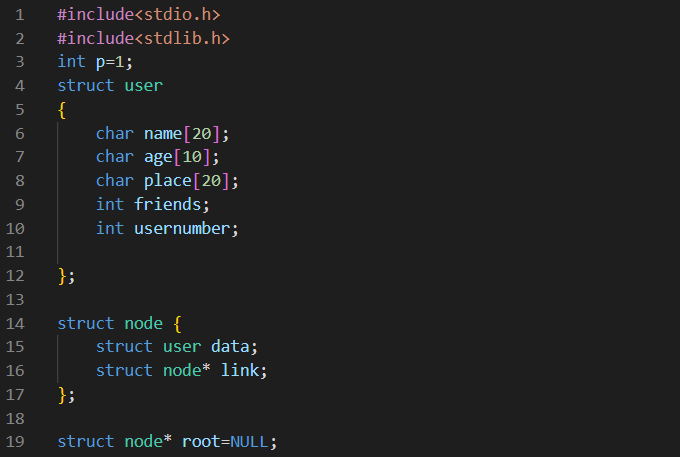
7.Total\_USERS

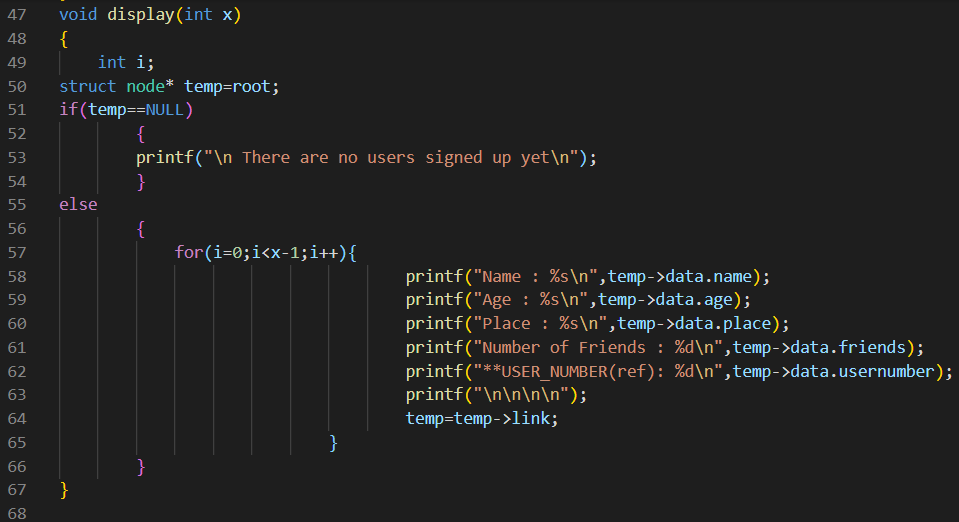
8.All USERS and Their friends

9.Common friends between two

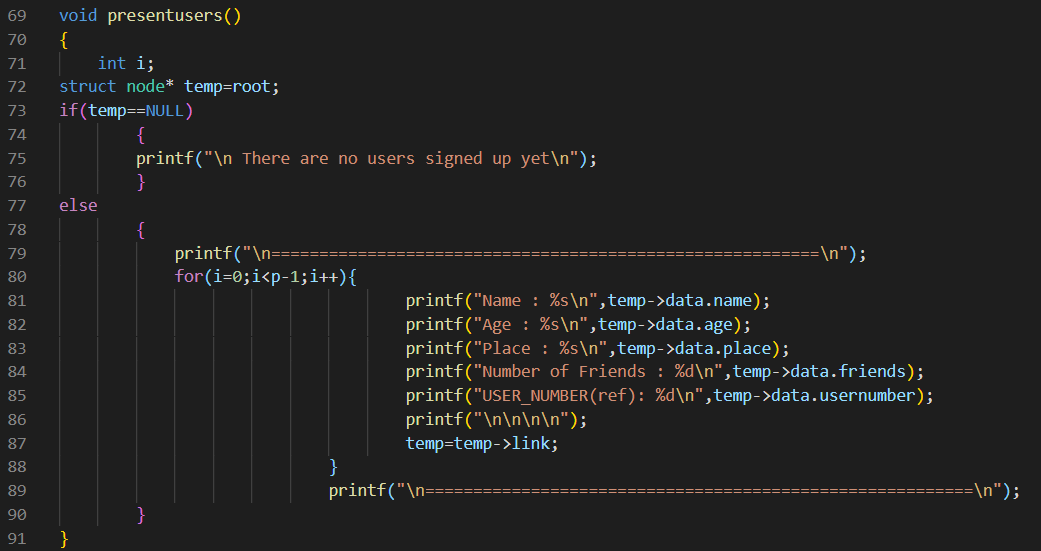
10.quit

SOLUTION CODE

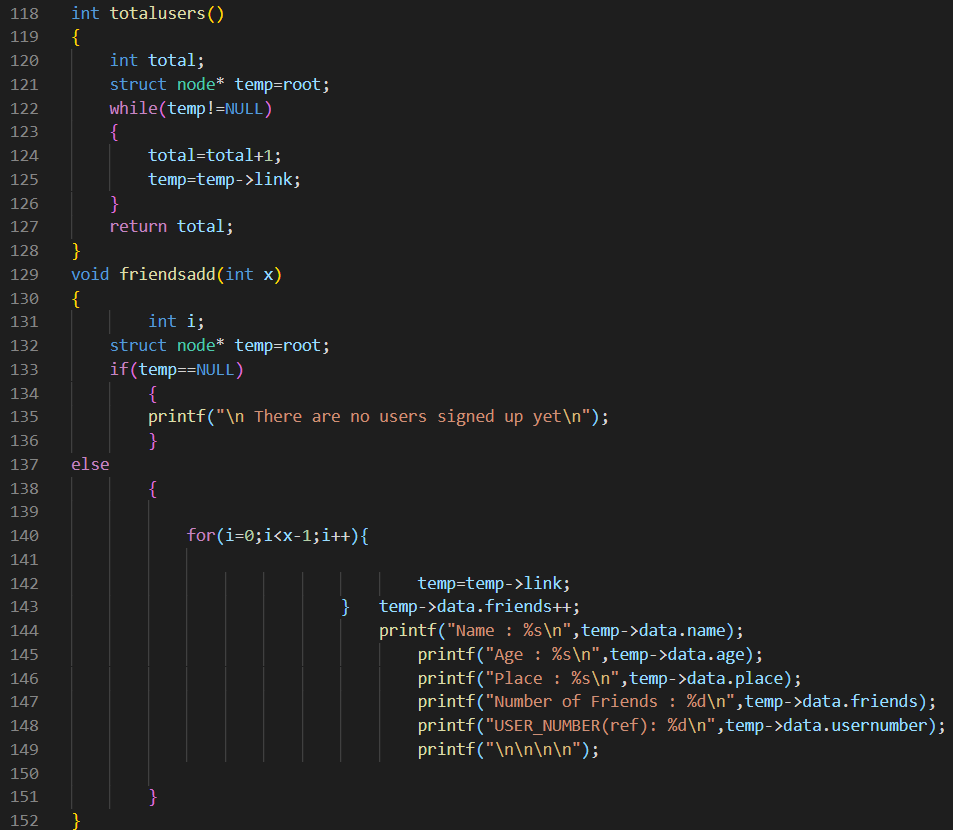


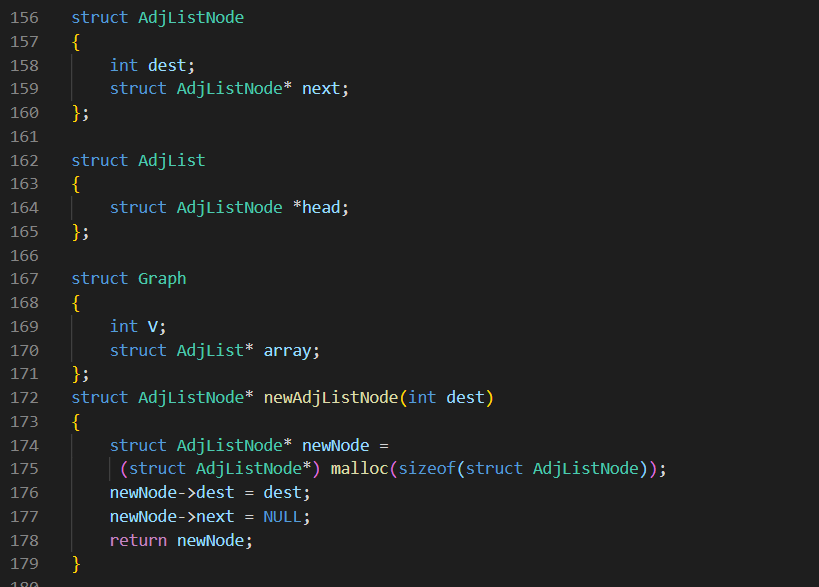


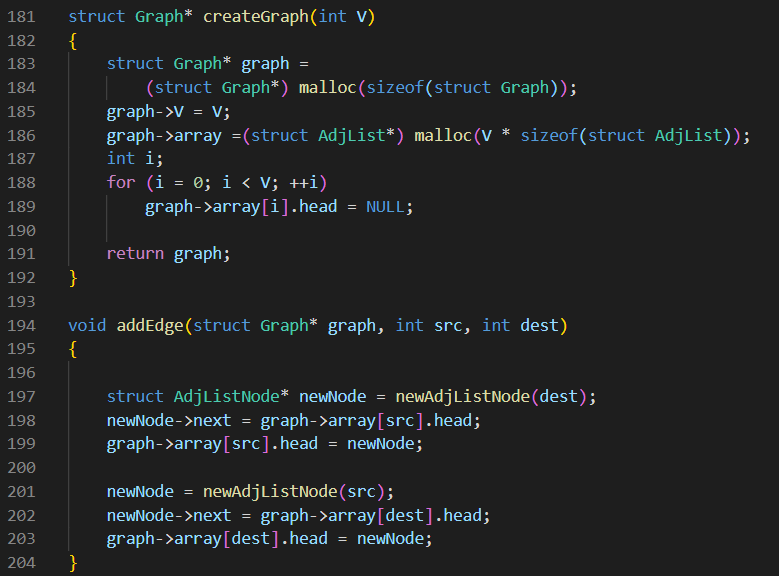


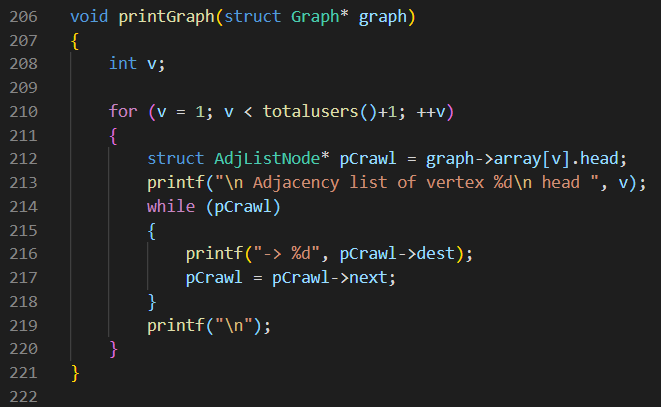


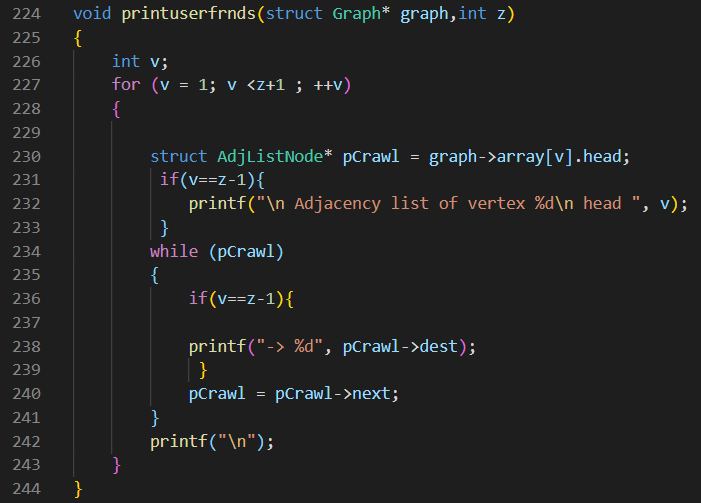


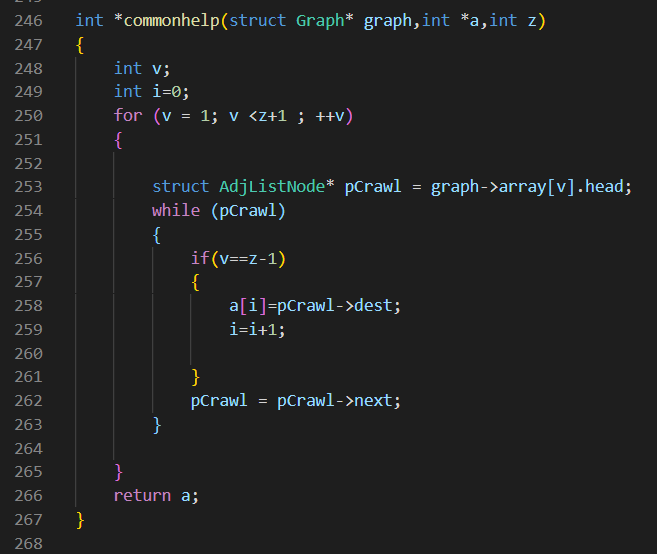


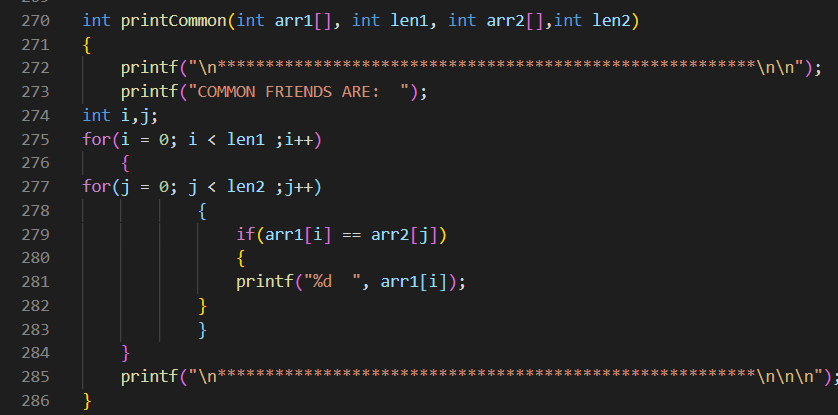




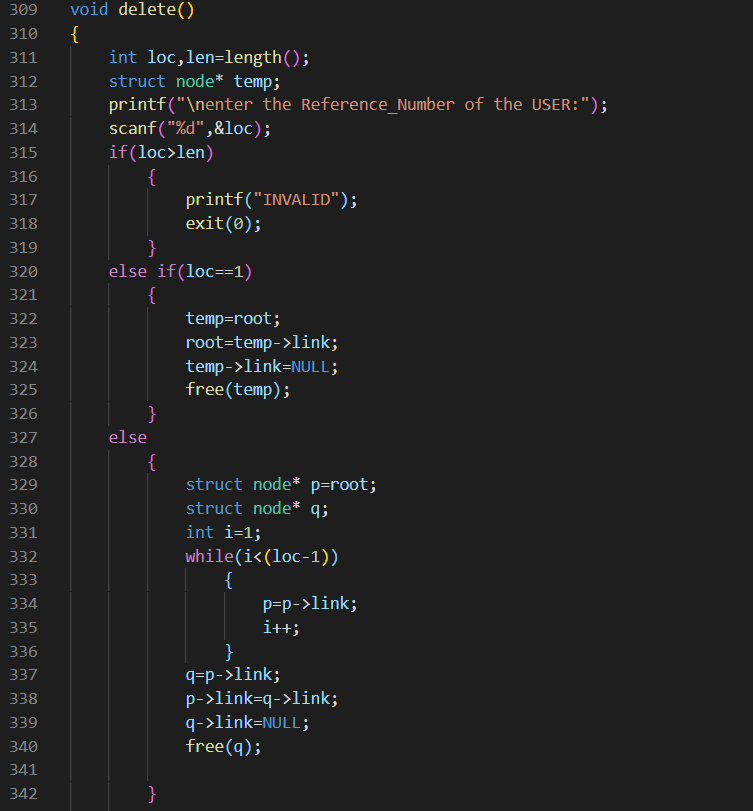


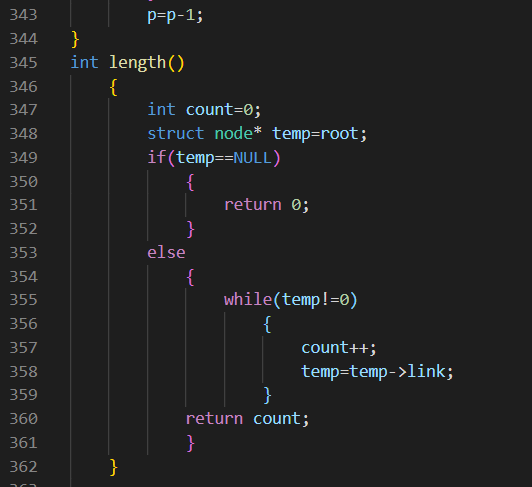


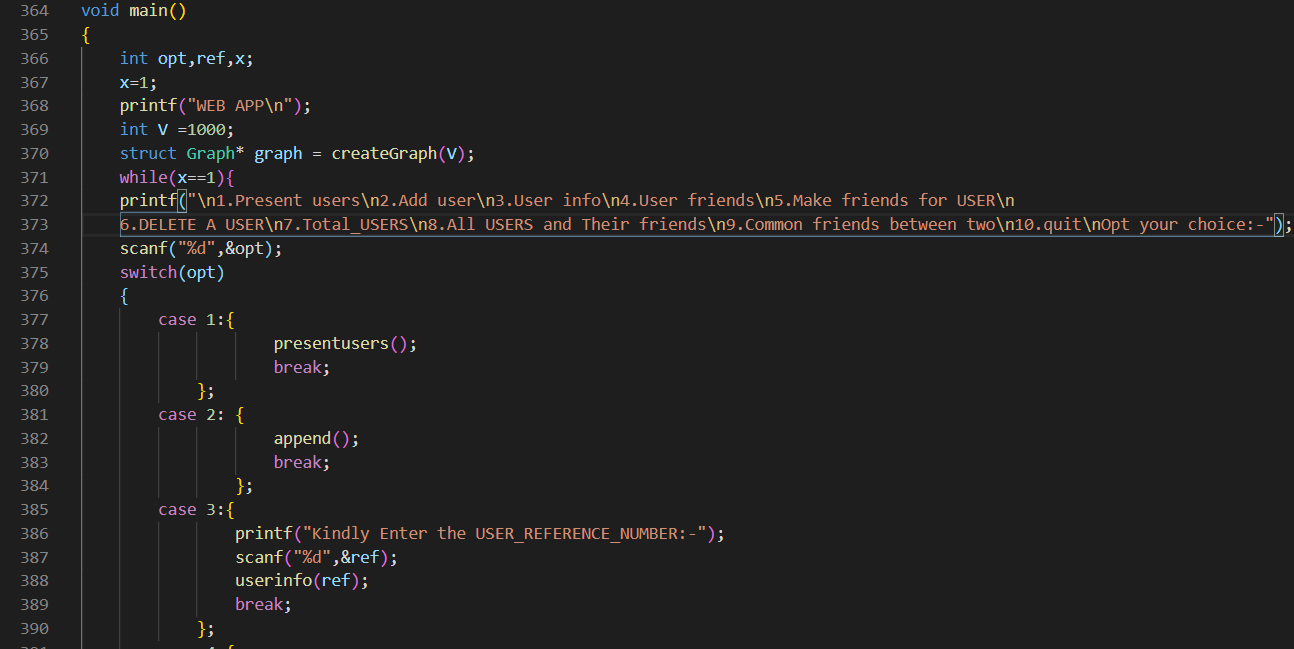


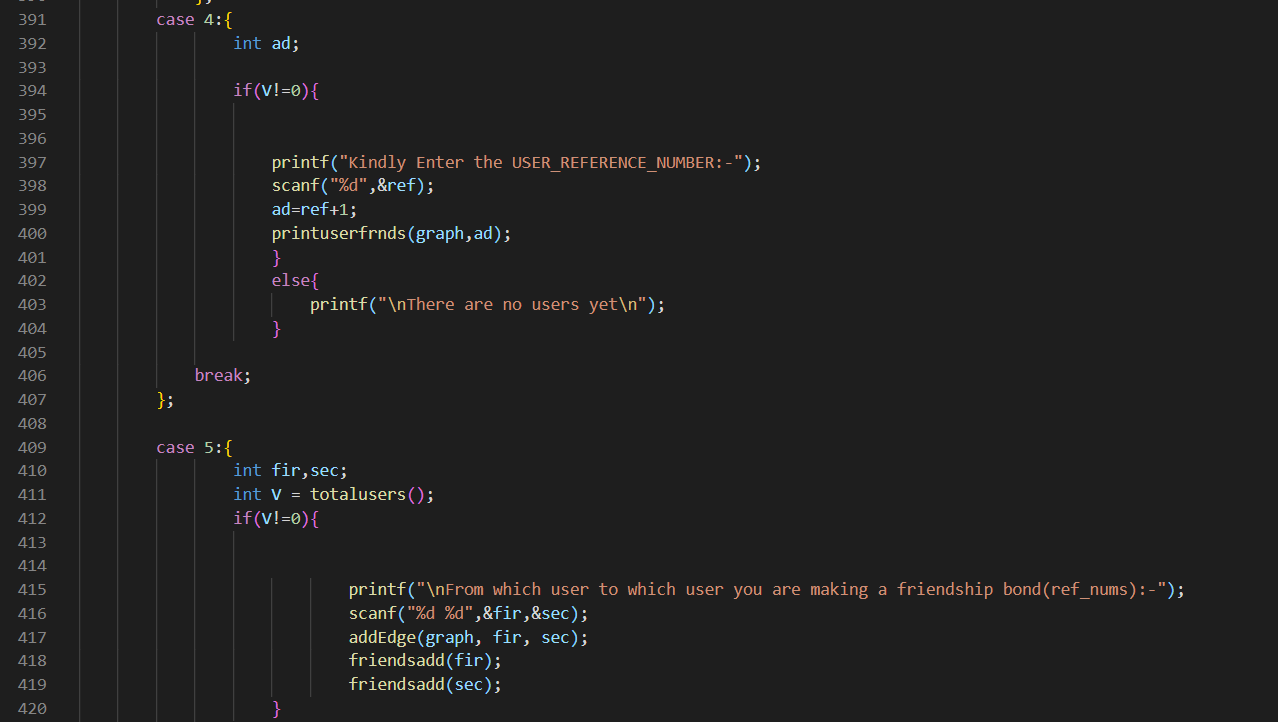


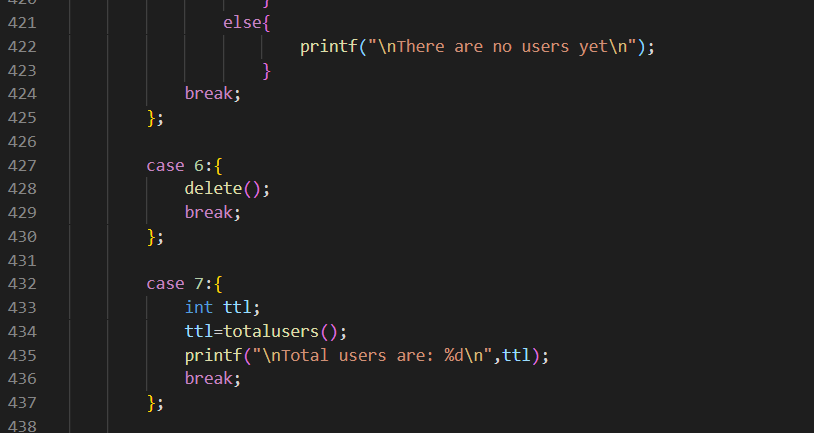


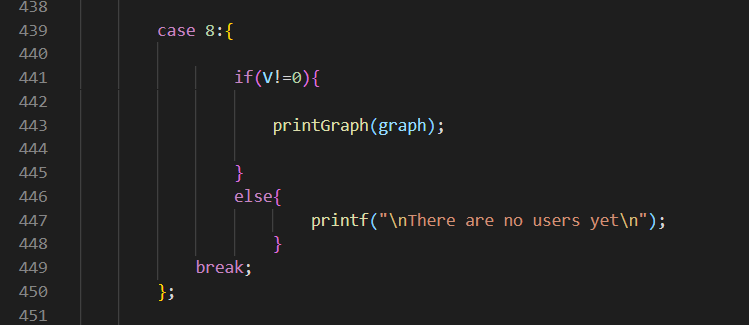
















SCREENSHOTS OF THE OUTPUT:

