Name: Tee Chee Guan StudentID:18202044

Software Engineering Project 1

Input

I used the function inputFacebookUsers from main.c to get input for the Facebook users, and that function calls another function from input.c called inputFriends which gets input for the friends of every Facebook users (max 4). The block of code which asks the user to input the number of the facebook user for which it should provide friend suggestions is implement in the main function in main.c as it is relatively small. Each function gets input until the newline character is entered or the maximum user/friends is reached.

The function inputFacebookUsers uses a switch statement to determine which user is currently selected and calls the function inputFriends with their respective friends array (friendsUser1..) as the first argument. The total number of friends of the current user is also stored in the numFriendsPerUser array according to their index.

The function inputFriends removes the trailing newline character(if any) from the user input (for Facebook users and Facebook friends). The function also takes the name of the current user as input to display it in the print statement. This function works quite similarly to inputFacebookUsers in general.

Sorting

The algorithm I used for the first part of the sorting process which is sorting the friend recommendations according to the number of friends theu have is Selection Sort. The reason is that according to the lectures and slides, selection sort seem to be more efficient than insertion sort and also easier to implement. Besides, I also get the chance to try out difference sorting algorithms. The second part which is sorting their names alphabetically if their friends count are equal is implemented using insertion sort as we only need to sort the users where their friends count are equal (partially sorted) and insertion sort can quickly sort a partially sorted array faster than selection sort (approximately $O(m \log n)$).

Time Complexity of selection sort is always $O(n^2)$ whereas insertion sort should perform better as its worst case complexity is $O(n^2)$. Generally it will take lesser or equal comparisons then $O(n^2)$.

This function sorts users depending on the number of their friends (from the biggest to the smaller). If 2 users have the same number of friends they will be sorted alphabetically. When sorting according to the number of their friends, selection sort is used for efficiency purposes. When sorting alphabetically, insertion sort is used as the array is already partially sorted (faster).

Additionaly, the function swaps the name of suggestions and also their friend count, in every swap.

Input:

suggestions - Number of recommended users

friendSuggestions - 2d array which stores the names of the suggested friends

numFriendsOfSuggestions - stores the friend count of the suggested friends

Friend Suggestions

To fill the friend suggestions arrays (friendsSuggestions and numFriendsOfSuggestions) of the selected user, I created a function getUserAndPopulate to fill the friend suggestions arrays (friendsSuggestions and numFriendsOfSuggestions) of the selected user, I created a function getUserAndPopulate which by using a switch statement calls the function populateArrays from recommendation.h and also parses the selected user's friends array as input (friendsUser1..).

This function populate the selected user with possiblefriendsSuggestions, it fills friendsSuggestions with the suggested names and also fills numFriendsOfSuggestions with the friend count of each suggested friend. It uses a nested loop to determine if the which users is not the friend of the selected user. (note that can't include the user himself).

Input:

totalUsers - total number of users (max 6)

user - is the index of the current selected user

users - 2d array which contains the names of all users

friendList - 2d array which contains the friends of the selected user

numbernumFriendsPerUser - friend count of each user

friendSuggestions - 2d array which stores the names of the suggested friends

numFriendsOfSuggestions - stores the friend count of the suggested friends

Output:

count - the total number of suggested friends