Software Engineering Project – Program Outline

The program starts with the implementation of the code given to us by the lecturer. After we input the Facebook User, main hands control over to the input.c function. Here I used a while loop that looped a max of 4 times and used the function 'fgets' to read input from the command line, as it is the easiest to implement which allows spaces to be inputted in the string. We return I so that numFriendsPerUser can be filled with the amount of friends that each user has.

The next function that gets called is choose.c. Essentially this function lets the user pick which Facebook user they want friend suggestions provided for. If they pick a number higher than the amount of users inputted (or 0 or lower), I make them choose again until they pick a valid number. I return their choice into the variable 'choice' in main.

Suggest.c is the next function called. This function places all possible friend suggestions into an array called friendSuggestions. No sorting is done here. We use a switch case to see which Facebook user we are dealing with, because the friendsUser array that we need to look through will depend on the Facebook user chosen. E.g. if the user picks Facebook user 2, we will need to look through the friendsUser2 array to see what Facebook users our chosen Facebook user is already friend's user. If the program finds a Facebook user who is not already friends with the user selected Facebook user, we add it to the friendSuggestions array. (N.B., I do not have a default case or my switch case, as it is impossible for the program to reach this switch case without the variable choice containing a value that passes one of my cases, therefore I would never hit the default case.) The last thing this function does is place the number of friends of each friendSuggestions in a new array, numFriendsSuggestions.

The last function we call is sort.c. In order to sort the friendSuggestions numerically, I use quicksort as it is one of the faster sorting algorithms available. We swap the items in numFriendsSuggestions, and then using the same index's, swap the strings in friendSuggestions, in order to keep the relationship between the two arrays the same. In order to sort alphabetically, I slightly modified the quick sort algorithm. We check to see if two users have the same number of friends (and that this value isn't zero). If this is true, we compare the strings alphabetically and swap them if the second one is alphabetically smaller etc. This will preserve the numerical order but order the users who have the same number of friends.

The last thing we do if print "There are no friend suggestions" if there are none, or else we print 1 (or 2 if available) friend suggestions for the selected user.