COMP10050: Software Engineering Project I
Assignment 1: Facebook Friend Suggestions
Design & Implementation

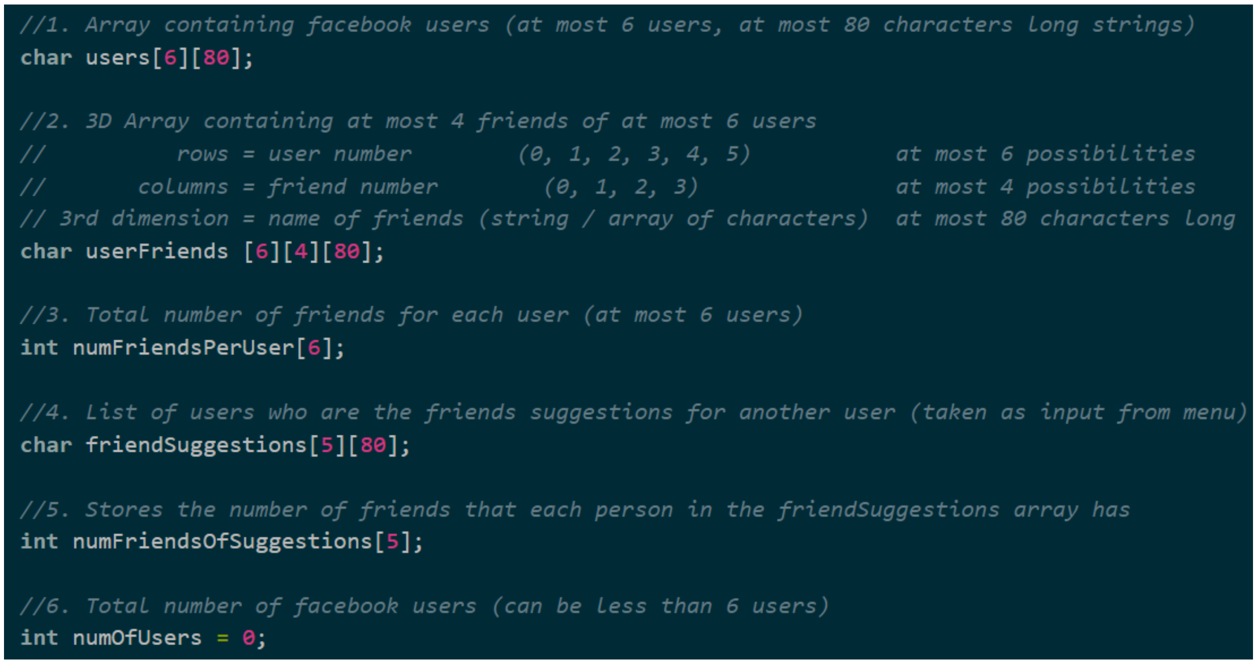
Rajit Banerjee
18202817

Source files: main.c, input.c, sorting.c
Header files: input.h, sorting.h

Design decisions:
How are friend suggestions selected? – shown under suggestFriends () function details below
How is the sorting algorithm implemented? – shown under sorting.c file details below

1. Global variables:

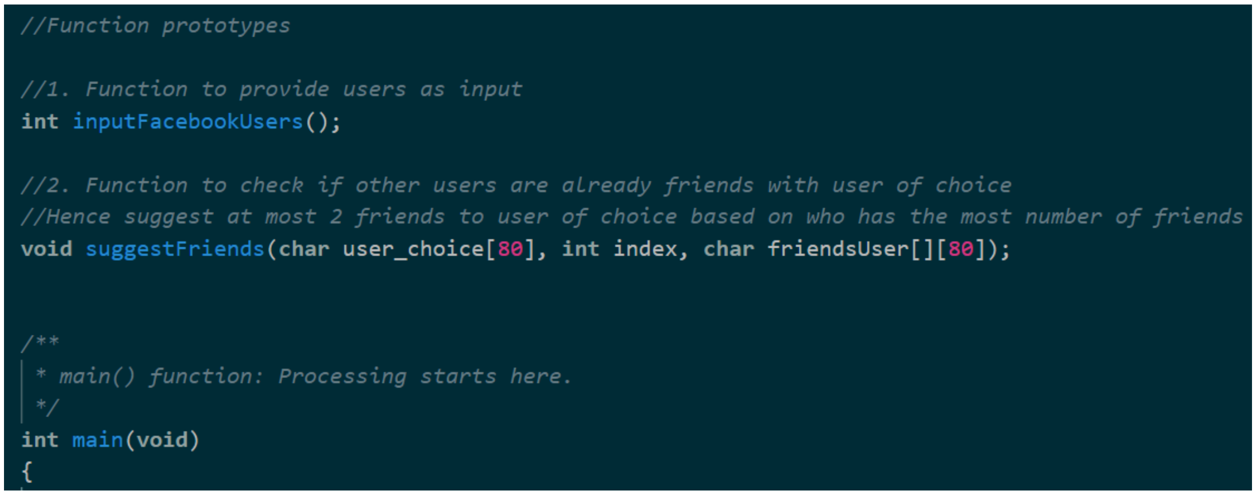
Storing lists of friends in 3D array (userFriends [6][4][80])
Instead of having 6 different arrays to store the names of friends for different users, a 3D array has been used. It has 6 rows to store friends for at most 6 users, and each user can have at most 4 friends (4 columns). The 3rd dimension represents the character array containing the actual name of a friend, which can be at most 80 characters.



```
//1. Array containing facebook users (at most 6 users, at most 80 characters long strings)
char users[6][80];

//2. 3D Array containing at most 4 friends of at most 6 users
//           rows = user number        (0, 1, 2, 3, 4, 5)          at most 6 possibilities
//        columns = friend number        (0, 1, 2, 3)              at most 4 possibilities
// 3rd dimension = name of friends (string / array of characters)  at most 80 characters long
char userFriends [6][4][80];

//3. Total number of friends for each user (at most 6 users)
int numFriendsPerUser[6];

//4. List of users who are the friends suggestions for another user (taken as input from menu)
char friendSuggestions[5][80];

//5. Stores the number of friends that each person in the friendSuggestions array has
int numFriendsOfSuggestions[5];

//6. Total number of facebook users (can be less than 6 users)
int numOfUsers = 0;
```

2. Functions in main.c file:

- int main(void)
- int inputFacebookUsers ()
- void suggestFriends (char [80], int, char [][80])



```
//Function prototypes

//1. Function to provide users as input
int inputFacebookUsers();

//2. Function to check if other users are already friends with user of choice
//Hence suggest at most 2 friends to user of choice based on who has the most number of friends
void suggestFriends(char user_choice[80], int index, char friendsUser[][80]);


/**
 * main() function: Processing starts here.
 */
int main(void)
{
```

(a) int main(void)

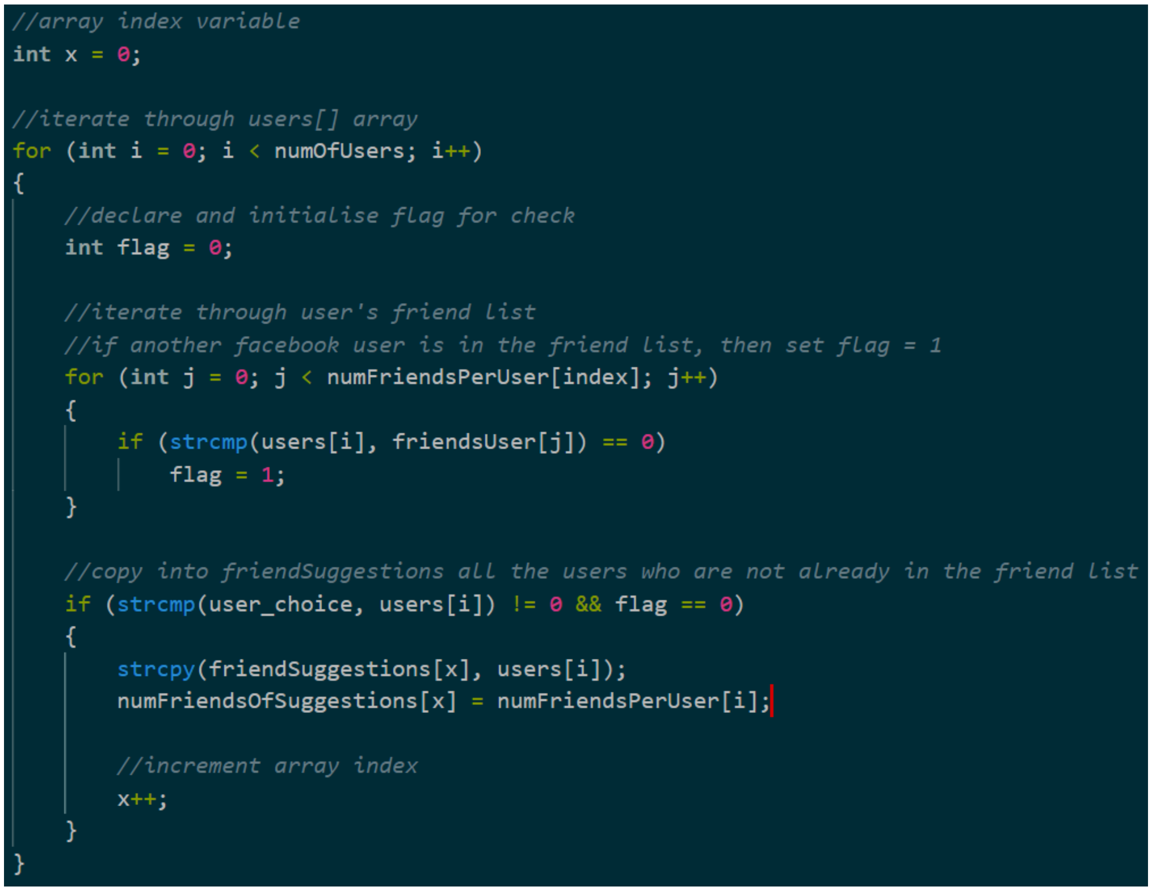
- Calls inputFacebookUsers () and assigns number of users entered to numOfUsers
- Displays the names of users entered
- Asks us to choose a user from the list and calls suggestFriends () function to suggest at most 2 friends to selected user

(b) int inputFacebookUsers ()

- **char** *fgets (**char** *__restrict__ __s, **int** __n, FILE *__restrict__ __stream)
  is the function used to take input from command line. This is used because we know where to store the entered data (character array), how long the maximum input length will be (80 characters), and also the fact that we're reading from file stdin (standard input, i.e., keyboard).
- Allows us to input at most 6 Facebook users and also calls function inputFriends () in input.c file in order to input at most 4 friends for each user.
- Also stores the number of friends entered for each user in numFriendsPerUser []

(c) void suggestFriends (char [80], int, char [][80]))

- In this function, we iterate through the list of Facebook users and check if they are already friends with the user for whom suggestions are required.
  The algorithm works by setting a flag = 1 if a match is found. In a particular iteration, if match is not found, then flag remains = 0, i.e., the current Facebook user is not part of the friend list of the chosen user (for suggestions). If that's the case (flag is 0), then the Facebook user is added to the list of suggested friends. It is also checked that the current iteration Facebook user is not the chosen user themselves.
  The flag is reset to 0 after each iteration so that the check can be performed on the next Facebook user.



```c
//array index variable
int x = 0;

//iterate through users[] array
for (int i = 0; i < numOfUsers; i++)
{
    //declare and initialise flag for check
    int flag = 0;

    //iterate through user's friend list
    //if another facebook user is in the friend list, then set flag = 1
    for (int j = 0; j < numFriendsPerUser[index]; j++)
    {
        if (strcmp(users[i], friendsUser[j]) == 0)
            flag = 1;
    }

    //copy into friendSuggestions all the users who are not already in the friend list
    if (strcmp(user_choice, users[i]) != 0 && flag == 0)
    {
        strcpy(friendSuggestions[x], users[i]);
        numFriendsOfSuggestions[x] = numFriendsPerUser[i];

        //increment array index
        x++;
    }
}
```

^ Fill array of friend suggestions and find how many friends each of them has.
(using flag concept)

- The function takes as parameter the name of the chosen user, the index (of the array users []) at which the said name is stored, and also the friend list of the chosen user.
- It calls the sort() function in sorting.c file in order to sort the suggested friends alphabetically and in decreasing order of number of friends
- Finally, it displays at most 2 friend suggestions, i.e., elements at indices [0] and [1] are printed if they are not empty

3. Function in input.c file (header file – input.h):

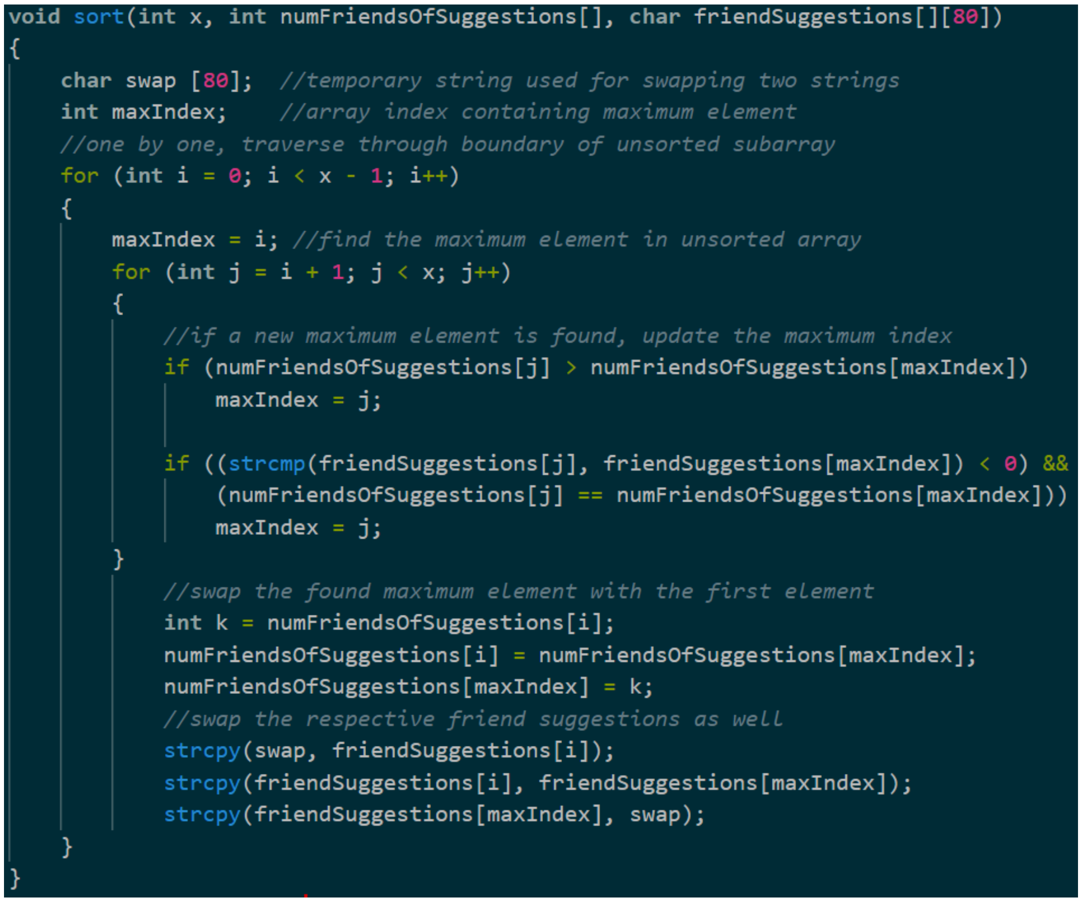
- **int inputFriends (char friends [][80], char user_name [80])**
  This function inputs (into friends [][80] array) at most 4 friends for the current Facebook user (whose name is stored in user_name [80]). It also returns the number of friends provided as input for current user.

4. Function in sorting.c file (header file – sorting.h):

- **void sort (int, int [], char [][80])**
  This is the function where the selection sort algorithm is implemented, which is used because it's more efficient compared to bubble sort.

Selection Sort Algorithm (descending order)

```
1. Get a list of unsorted numbers
2. Set a marker for the unsorted section at the front of the list
3. Repeat steps 4 - 6 until one number remains in the unsorted section
4.     Compare all unsorted numbers in order to select the largest one
5.     Swap this number with the first number in the unsorted section
6.     Advance the marker to the right one position
7. Stop
```



```c
void sort(int x, int numFriendsOfSuggestions[], char friendSuggestions[][80])
{
    char swap [80];  //temporary string used for swapping two strings
    int maxIndex;    //array index containing maximum element
    //one by one, traverse through boundary of unsorted subarray
    for (int i = 0; i < x - 1; i++)
    {
        maxIndex = i; //find the maximum element in unsorted array
        for (int j = i + 1; j < x; j++)
        {
            //if a new maximum element is found, update the maximum index
            if (numFriendsOfSuggestions[j] > numFriendsOfSuggestions[maxIndex])
                maxIndex = j;

            if ((strcmp(friendSuggestions[j], friendSuggestions[maxIndex]) < 0) &&
                (numFriendsOfSuggestions[j] == numFriendsOfSuggestions[maxIndex]))
                maxIndex = j;
        }
            //swap the found maximum element with the first element
            int k = numFriendsOfSuggestions[i];
            numFriendsOfSuggestions[i] = numFriendsOfSuggestions[maxIndex];
            numFriendsOfSuggestions[maxIndex] = k;
            //swap the respective friend suggestions as well
            strcpy(swap, friendSuggestions[i]);
            strcpy(friendSuggestions[i], friendSuggestions[maxIndex]);
            strcpy(friendSuggestions[maxIndex], swap);
    }
}
```

In this function, the suggested friends are sorted in decreasing order by the number of friends each of them has. Also, if two suggestions have the same number of friends, then their positions are swapped to have them sorted lexicographically (increasing alphabetical order).