DEVELOPER GUIDE
MUHAMMAD HAMEEZ KHAN (TFBB32)

FILE HEADERS USED IN THE PROGRAM:

- #include<stdio.h>
- #include<stdlib.h> (Used for dynamic memory allocation)
- #include<string.h> (Used for operations related to strings)
- #include<ctype.h> (Used for a function tolower)

STRUCTURES USED IN THE PROGRAM:

Date: stores the date of birth of individual user.

member: stores the data of an individual user.

statistics: stores the data of all users in an array and their count.

FILE HANDLING FUNCTIONS:

- **void read_from_file(statistics * data);**

Parameters: a pointer pointing to statistics structure.

Operations: opens a file inside the function and creates a buffer array that reads the data line by line from the file

of single member at a time and stores it in the memory inside the statistics structure

To store a member a dynamic structure is created which stores the data of the member with the help of string tokenization function which returns a string of given starting point to the given stopping parameter and strdup converts it to dynamic string. To read the integer values strtol function is used which converts the string to long int and then it is converted to int.

File is closed in the end

Call: the function is called in main.

Return: The function returns nothing.

- **void print_in_file(statistics *data);**

Parameters: a pointer pointing to statistics structure.

Operations: opens a file inside a function in write mode and starts printing the data in a file

The data of each member is stored line by line.

if the user is in no group then the first condition of if statement is executed

If the user is in a group then the else is executed

The file is closed in the end

Call: the function is called in main.

Return: The function returns nothing.

FUNCTIONS USED IN THE PROGRAM:

- **member* add_new(statistics * data);**

Parameters: a pointer pointing to statistics structure.

Primary Purpose: create a new member for the database.

Operation: Dynamically creates a new member structure and receives input from the user one by one for every pointer and dynamically allocates memory for every string and checks if the string name is valid. If it is invalid then the string memory is freed, and a new name is asked from the user. The nickname of each user is unique and then it is checked in the function if the nick name entered by the user already exists or not in the database. If it already exists, then the nickname is freed and a new nickname is asked from the user. After all the data is collected from the user and stored in the structure then the function returns the user.

Call: the function is called in main.

Return: returns the data of one user.

- **date add_date();**

Parameters: No parameters.

Primary purpose: creates a date structure for a member.

Operation: ask the user for date input and prints an invalid date error message if the date entered by the user is incorrect and will ask for another date as long as the user does not enter a correct date and then returns it.

Call: the function is called in add_new(statistics * data) function.

Return: returns the date of a user

- **int invalid_date(date tmp);**

Parameters: a copy of date type which was given by user as input.

Primary purpose: checks whether a user has enter correct date or not

Operation: It individually checks the year, month and day in the given date structure and checks if they are negative or zero or nonvalid numbers using the comparision operators and returns 1 if the date is invalid or 0 if it is valid.

Call: the function is called in add_date() function.

Return: returns an integer (1 if incorrect and 0 if not)

- **int leap_year(int year);**

Parameters: a copy of int year which was given by user as input

Primary purpose: checks whether the year is leap year or not.

Operation: A year is a leap year if it is divisible by four and not divisible by 100 or if it is divisible by 400. The functions uses some comparision operators and if one the two conditions is true then the function returns 1 (indicating it is a leap year) otherwise 0(indicating not a leap year).

Call: the function is called in invalid_date(date tmp) function.

Return: returns an integer (1 if incorrect and 0 if not)

- **char* add_name();**

Parameters: no parameters.

Primary purpose: dynamically allocates the memory for the strings with the exact size needed and having no limitations on the length of users input.

Operation: dynamically allocates the memory for the strings. It is used for all strings (nickname,familyname,givenname,place). The function reads an input stream character by character in a while loop as long as it is successful and it is not a new line and dynamically allocates a new array size copy all the characters from the old array to the new array add the new character in the end of the new array, frees the old array and copies the address of the new array in the old array.

Call: this function is the most utilized function it was used in the following functions:

search_member(statistics data)

add_group(int *total,char*** group)

add_new(statistics * data)

delete_member(statistics data)

print_group_members(statistics data)

add_member_in_group(statistics * data)

delete_member_in_group(statistics * data)

Return: the function returns a dynamically allocated string with the exact size needed for the string.

- **void print_single_user(member profile);**

Parameters: a copy of member structure which must be printed.

Primary purpose: prints a single member of the database.

Operation: prints the data of the given member.

Call:

The function is called in two functions.

search_member(statistics data)
print_group_members(statistics data)

- **statistics add_in_list(member * to_add, statistics old);**

Parameters: a pointer pointing to the structure that holds the data of the new user that we have to store in the dynamic array of structure

A copy of the statistics structure.

Primary purpose: adds a member in the database.

Operation:

Creates a new dynamic array storing member structure with the size of the count of all the existing data and adding one for the new member that we have to store and then copying all the data from the old array into the new array and then adding the new member data(structure) into the end of the array and then freeing the old array and then storing the address of the new array into pointer that is in the statistics structure and increasing the count by one and then returning the new statistics structure.

Call:

read_from_file(statistics * data): the read_from_file function reads the data of the users one by one and stores the data with the help of add_in_list function.

Main function: using this in the main function allows the user to store the data of a new user.

Return: returns a statistics structure with new count and a new dynamic array.

- **statistics delete_member(statistics data);**

Parameters: A copy of the statistics structure.

Primary purpose: deletes a member in the data.

Operation:

Initially asks the user for the nickname of the user to be deleted with the help of add_name() function and then looping through the array with for loop comparing the nickname given by the user with nicknames of the users stored in the data (with strcmp function). If the nickname is matched with a user then the dynamic memory storing the users data is freed including the pointers inside group array pointing to strings and then a new dynamic memory of one less size is allocated, and the old data is copied from the old array to the new array using a for loop using j and k variable. J variable for the new array and k for the old variable. Incrementing both j and k after every index is copied from old to new and if k is on the index of the deleted user (if k == i) then we want to increment k by 1 because we don't want to copy anything from that index as everything is freed from that index. After copying all the data, we free the old array and decrease the total size count by 1 and since the nickname of each user is unique then a break statement is used to exit the loop and then the nickname string is freed which was used for comparison. If we find no user with the given nickname, then the if statement is never executed, and the no user is deleted.

Call: the function is called in main function.

Return: the function returns a new statistics structure

- **void search_member(statistics data);**

Parameters: a copy of statistics structure

Primary purpose: search a member in the database

Operation:

The user is given two choices to search for a member. The first one is to search by the family name and the second choice is to search by the nickname.

The users selects among the two choices given to search for a member. If the user presses one then the user is asked to enter a family name which is stored in a family_name pointer and then the given family name is compared by using for loop for traversing through the array. The users with same family name are printed, after the loop family_name pointer is freed.

If the user presses one then the user is asked to enter a nick name which is stored in a nick_name pointer and then the given nick name is compared by using for loop for traversing through the array. The user with same nick name is printed and then nick_name pointer is freed.

Call: the function is called in main.

Return: the function returns nothing

- **int improper_name(char * check);**

Parameters: a pointer pointing to a dynamically allocated string.

Primary purpose:

The improper_name function is designed to check if a given string contains characters other than letters (both uppercase and lowercase).

Operation:

It uses a for loop to iterate through each character in the string until it encounters the null terminator ('\0'), which marks the end of the string.

Within the loop, it checks if the current character is not a letter (either uppercase or lowercase). The condition !(check[i] >= 'A' && check[i] <= 'Z') && !(check[i] >= 'a' && check[i] <= 'z') checks if the character is not within the range of uppercase letters and not within the range of lowercase letters.
If the character is not a letter, it frees the memory allocated for the string using free(check) and returns 1, indicating that the name is improper.
If the loop completes without finding any non-letter characters, the function returns 0, showing that the name is proper.

Call: the function is called in add_new();

Return: the function returns integer.

- **void add_group(int *total,char*** group);**

Parameters:

int *total: Pointer to an integer that will store the total number of groups.

char*** group: Pointer to a pointer that is pointing to array of pointers pointing to a char, which represents a 2D array to store group names.

Primary purpose:

The add_group function dynamically allocates memory to store groups of a member(the names of group which the member have joined).

Operations:

The function first asks the user to input the total number of groups they want to add.
It reads this value and stores it in the variable pointed to by total.
then dynamically memory is allocated for an array of pointers to char by the command **(char**)malloc(sizeof(char*) * *total)** to store group names.
The group names are asked by the user which runs *total times and dynamically allocates memory for each group name using add_name() function and stores it in the pointers in group array and then the group array is stored in the *group (pointer pointing to double pointer).
Call: the function is called in add_new() function
Return: the function returns nothing as the data is stored in addresses the pointers are pointing to.

----

void print_group_members(statistics data);

Parameters:

A structure containing the array of members and the total number of members.

Primary purpose: to print the data of members that are in the given group.

Operation:

The function first asks the user to enter a group name using the add_name() function. This dynamically allocates memory for the group name.

It then uses nested loops to iterate over all members and their groups in the database.
The outer loop i goes through the member array that is pointed by the pointer.

The inner loop j iterates over all groups of the current member and checks if the current group name matches the user input group name using strcmp.
If a match is found print_single_user() function is called to display the details of the member.
The break statement is used because we further don't need to check the groups of that member that is stored in the array as we already found it.
After the nested loop we free the group name that was given by the user to prevent memory leak.
Return: the function returns nothing as we are not adding or removing anything from the data.

- **void add_member_in_group(statistics * data);**

Parameters: a pointer pointing to statistics data structure

Primary purpose: The function is responsible for adding a member to a specific group. It asks the user to enter the nickname of the member and the name of the group. If the member exists, and the group doesn't already contains that member, it adds the member to the group.

Operation:

The functions first asks the nickname of the user which we have to add in a group and then the group name.

A for loop is used to iterate all the members of the group to find the member with the entered nickname

If the member is found, it checks if the member is already in the specified group with another for loop
If the member is found and is not already in the group, it calls the adding_in_group function which adds the member in the group.

If the member is not found, a message is printed that no member with the given nickname exists.
If the member is already in the group a message is printed that the member is already in the group.

Call: the function is called in the main function.

Return: the function returns nothing.

- **void adding_in_group(char * group_to_add, char*** group_array, int * total);**

Parameters:

char *group_to_add: a pointer pointing to the name of the group to be added.
char ***group_array: A pointer to a pointer to an array of pointers pointing to strings which are the existing groups the member has already joined.
int *total: A pointer to an integer which is the total number of groups.

Primary purpose:

The function adds a new group to the list of groups associated with a member.

Operations:

First It dynamically allocates memory for a new array of strings with size+1 to hold the updated group names by using **char **new = (char**)malloc((size+1) * sizeof(char *));**

It copies the existing group names from the old array to the new array using for loop.
It adds the new group name to the end of the new array.

It frees the memory occupied by the old group array.
using free(*group_array);
It updates the pointer to the group array to point to the newly allocated memory(*group_array = new;) and increments the total number of groups.(*total = size + 1

Call: the function is called in add_member_in_group() function.

Return: the function returns void.

- **void delete_member_in_group(statistics * data);**

Parameters:

A pointer, pointing to the statistics structure

Primary purpose:

This function is designed to remove a member from a specific group. It prompts the user to enter the nick name of the member and the group name from which they should be removed. It then searches for the member in the data, finds the group, and deletes the member from that group.

Operations:
The function asks the user for the group name(group)and the nickname of the member (nick_name) which we have to remove using the add_name function.

It iterates over all members in the data structure to find the member with the specified nick name (nick_name).
If the member is found, it proceeds to check if they are part of the specified group (group).

If the member is found in the group, it frees the memory allocated for the group name in the member's data and calls the deleting_in_group function which deletes the member from the group and then frees the group and nick_name string.
If the member is already not in the group a message is printed that the member is not in the group and group and nick_name memory allocation are freed.

If no user with the given nickname exists then the whole function will iterate the whole data and print a message in the end that the no user with the given nickname exists and then group and nick_name memory allocation are freed.

Call: the function is called in main function.

Return: the function returns void.

- **void deleting_in_group(int index, char*** group_array, int * total);**

Parameters:

int index: The index of the element to be deleted from the group_array.
char*** group_array: A pointer pointing to a pointer pointing to an array of pointers pointing to strings(array of characters), representing a pointer to the array of group names.
int * total: A pointer to an integer representing the total number of groups.

Primary purpose: This function handles removing an element at a specified index from an array of group names. It adjusts the array and updates the total number of groups accordingly.
Operations:

It stores the current total in the variable size.

We have the index at which we had the deleted group.
It dynamically allocates memory for a new array (new) with a size of (size-1) where size is the current total number of groups.
It iterates over the original array (*group_array) and copies elements to the new array, excluding the element at the specified index at which the user was deleted. It uses two indices (**i** for the new array, **j** for the old array) to skip the element at the specified index.
It frees the memory occupied by the original array using free(*group_array)
It updates the pointer *group_array to point to the new array.
It updates the value pointed to by total to reflect the new total number of groups (size - 1).

Call: the function is called in delete_member_in_group function.

Return: the function returns void.

- **Int main()**

Parameters: the function has no parameters

Operation:

It initializes a variable called choice which will be used in switch case to exit a certain case, including the statistics structure (data) to store member information.
It reads existing data from a file using the read_from_file function and stores it in statistics structure (data).
The function executes a do while loop which prints the menu and uses a switch statement to execute different tasks based on the user's choice. The user can select among 8 choices and the loop ends when

the user selects option 8.If the user selects any other number other than 1 to 8 then an error message is printed.

Case 1 creates a new member and adds it in the statistics structure(data)

Case 2 searches a member in the statistics structure(data)

Case 3 deletes a member from the statistics structure(data)

Case 4 adds a member of statistics structure(data) in a particular group.

Case 5 deletes/remove a member of statistics structure(data) from a group.

Case 6 prints all the members of a particular group

Case 7 prints all the members stored in statistics structure(data)

Case 8 saves the data in a file, frees all the dynamically allocated memory used for storing all the members data in statistics structure(data) and prints a good bye message

After the end of loop the program returns 0, indicating successful program execution.

Return: the function returns int number 0 representing the code had no errors and any other representing the code exited with an error