PROGRAM TEST SCRIPT, ROMAN

loadUsers();

Function Description:
- This function loads the users struct with values from users.txt

Prototype: loadUsers(user users, int *userCount);

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Users.txt exists	users.txt	Data is transferred	Data is transferred	P
2	Users.txt does not exist	users.txt	No data is transferred	No data is transferred	P
3	Users.txt exists and has out-of-bound values	users.txt	Initialization of users is disorganized	Initialization of users is disorganized	P

saveUsers();

Function Description:
- This function saves all of the user's values to users.txt

Prototype: saveUsers(user users, int userCount);

Pre-Condition: N/A

Return:
- This function does not return anything; void;

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Users structure variable exists	Users struct variable	Data is transferred	Data is transferred	P
2	Users structure	Users struct	No data is	No data is	P

	variable does not exist	variable	transferred	transferred	
3	Users structure has out-of-bound values	Users struct variable	Saving of user data is disorganized	Saving of user data is disorganized	P

loadAdmin();

Function Description:
- This function loads the admin struct with values from admin.txt

Prototype: loadAdmin(administrator *admin);

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	admin.txt exists	admin.txt	Data is transferred	Data is transferred	P
2	admin.txt does not exist	admin.txt	No data is transferred	No data is transferred	P
3	admin.txt exists and has out-of-bound values	admin.txt	Initialization of admin is disorganized	Initialization of admin is disorganized	P

saveAdmin();

Function Description:
- This function saves the admin's values to admin.txt.

Prototype: saveAdmin(administrator admin)

Pre-Condition: N/A

PROGRAM TEST SCRIPT, ROMAN

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Admin structure variable exists	Admin struct variable	Data is transferred	Data is transferred	P
2	Admin structure variable does exists	Admin struct variable	No data is transferred	No data is transferred	P
3	Admin structure exists and has out-of-bound values	Admin struct variable	Initialization of admin is disorganized	Initialization of admin is disorganized	P

loadMessages();

Function Description:

- This function loads all the message values from msgs.txt.

Prototype: loadMessages(message msgs, int *msgCount)

Pre-Condition: N/A

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	msgs.txt exists	msgs.txt	Data is transferred	Data is transferred	P
2	msgs.txt does not exist	msgs.txt	No data is transferred	No data is transferred	P
3	msgs.txt exists and has out-of-bound values	msgs.txt	Initialization of msgs is disorganized	Initialization of msgs is disorganized	P

PROGRAM TEST SCRIPT, ROMAN

saveMessages();

Function Description:
- This function saves all the message values to msgs.txt

Prototype: saveMessages(message msgs, int msgCount);

Pre-Condition: N/A

Return:
- This function does not return anything; void;

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Msgs structure variable exists	Msgs struct variable	Data is transferred	Data is transferred	P
2	Msgs structure variable does exists	Msgs struct variable	No data is transferred	No data is transferred	P
3	Msgs structure exists and has out-of-bound values	Msgs struct variable	Initialization of msgs is disorganized	Initialization of msgs is disorganized	P

controlAccess();

Function Description:
- This function determines which menu the user should be in.

Prototype: controlAccess(int loggedIn, int exitApp);

Pre-Condition: N/A

Return:
- This function does not return anything; void;

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Valid inputs	loggedIn = 1, exitApp = 0	Transfer user to user menu	Transfer user to user menu	P

2	Invalid Inputs	loggedIn = 5, exitApp = -13	Exit app by default	Exit app by default	P

userLogin();

Function Description:
- This function calculates which user is trying to login.

Prototype: userLogin(user users, int userCount, int *userIndex, int *loggedIn);

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	A non-deleted user's userIndex is passed	userIndex = 0	User 0 logs in	User 0 logs in	P
2	A deleted user's userIndex is passed	users[userIndex].isDeleted = 1	Login function exits	Login function exits	P
3	No user in the system	userCount = 0	Login function exits	Login function exits	P

createAccount();

Function Description:
- This function creates accounts in the system.

Prototype: createAccount(user users, int *userCount, administrator admin)

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through

pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Maximum userCount is passed	userCount = 30;	Print “Maximum users reached” and return	Print “Maximum users reached” and return	P
2	Valid Inputs	userCount < 30 and users is passed	Create new account and increment userCount	Create new account and increment userCount	P
3	Admin structure variable and users structure variable do not exist	No admin and msgs in parameter	Function does not run	Function does not run	P

selectUser();

Function Description:

- This function calculates which user is trying to login.

Prototype: selectUser(user users, int userCount);

Pre-Condition: userCount must not be 0.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted user exists in the users array	users[0].isDeleted = 1	Print deleted status and prevent selection	Print deleted status and prevent selection	P
2	Users array and userCount are matched	Users[4] exists and userCount = 5	Print all valid users	Print all valid users	P
3	Users array and userCount are not	Users[2] exists and	Not all users can be	Not all users can be printed and	P

	matched	userCount = 1	printed and selected	selected	

viewProfile();

Function Description:
- This function views the profile of a user.

Prototype: viewProfile(user users, int userIndex, int userProfileID);

Pre-Condition: userProfileID must not be from a deleted user.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Profile not printed	Profile not printed	P
2	Valid user index is passed	users[userIndex].isDeleted = 0	Profile is printed	Profile is printed	P
3	User index with incomplete data is passed	users[userIndex].name = '\0'	Profile is printed except for incomplete data	Profile is printed except for incomplete data	P

browseProfile();

Function Description:
- This function provides multiple options to interact with a user.

Prototype: browseUsers(message msgs, user users, int userCount, int *msgCount, int userIndex);

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through

pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Profile not printed	Profile not printed	P
2	Users array and userCount are matched	Users[4] exists and userCount = 5	Print all valid users	Print all valid users	P
3	Users array and userCount are not matched	Users[2] exists and userCount = 1	Not all users can be printed and selected	Not all users can be printed and selected	P

sendMessage();

Function Description:
- This function creates personal messages, group messages, and announcements.

Prototype: sendMessage(user users, int userCount, int userIndex, message msgs, int *msgCount, int recieverIDArr[], int recieverCount);

Pre-Condition: recievrIDAr[] must not contain the ID of a deleted user.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Receiver count does not match the amount of IDs in the receiver ID array.	recieverCount = 2, recieverIDArr[ ]3	Not all recipients receive the message	Not all recipients receive the message	P
2	Deleted User index is passed	users[userIndex].isDeleted = 1	Message does not send	Message does not send	P
3	msgCount does	msgCount =	Message	Message gets	P

	not match with amount of messages in msgs[]	1, msgs[5]	gets lost in msgs.txt	lost in msgs.txt	

messageModule();

Function Description:
- This function determines which parameters to send to sendMessage(); and it also serves as a message menu.

Prototype: messageModule(message msgs, user users, int userCount, int *msgCount, int userIndex)

Pre-Condition: N/A

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Message parameters are not passed	Message parameters are not passed	P
2	Valid parameters are passed	userIndex = 1, userCount = 3, msgs = 5, msgCount = 5	Correct message parameters are passed	Correct message parameters are passed	P
3	Duplicates in recieverIDArr	recieverIDArr[0] = 1, recieverIDArr[1] = 1,	Duplicates are disregarded and correct message parameters are passed	Duplicates are disregarded and correct message parameters are passed	P

sendMessage();

Function Description:
- This function creates personal messages, group messages, and announcements.

PROGRAM TEST SCRIPT, ROMAN

Prototype: sendMessage(user users, int userCount, int userIndex, message msgs, int *msgCount, int recieverIDArr[], int recieverCount);

Pre-Condition: recievrIDAr[] must not contain the ID of a deleted user.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Message cannot be sent	Message cannot be sent	P
2	Maximum messages in system is reached	msgCount = 1000	Prevent messaging and return	Prevent messaging and return	P
3	Sender and receiver are the same	senderID = 1, recieverID[1] = 1;	Message cannot be sent	Message cannot be sent	P

messageModule();

Function Description:

- This function determines which parameters to send to sendMessage(); and it also serves as a message menu.

Prototype: messageModule(message msgs, user users, int userCount, int *msgCount, int userIndex)

Pre-Condition: N/A

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Message parameters are not passed	Message parameters are not passed	P
2	Valid parameters are passed	userIndex = 1, userCount =	Correct message	Correct message parameters are	P

		3, msgs = 5, msgCount = 5	parameters are passed	passed	
3	Duplicates in recieverIDArr	recieverIDArr[0] = 1, recieverIDArr[1] = 1,	Duplicates are disregarded and correct message parameters are passed	Duplicates are disregarded and correct message parameters are passed	P

printMessage();

Function Description:
- This function prints the details of a message.

Prototype: printMessage(message msgs, user users, int userIndex, int msgIndex, int checkingType);;

Pre-Condition: recievrIDAr[] must not contain the ID of a deleted user.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted user exists in the users array	users[0].isDeleted = 1	Print deleted status and prevent selection	Print deleted status and prevent selection	P
2	Users array and userCount are matched	Users[4] exists and userCount = 5	Print all valid users	Print all valid users	P
3	Users array and userCount are not matched	Users[2] exists and userCount = 1	Not all users can be printed and selected	Not all users can be printed and selected	P

isMsgRelated();

Function Description:
- This function determines the relation of a message to a user.

PROGRAM TEST SCRIPT, ROMAN

Prototype: isMsgRelated(user users, message msgs, int msgIndex, int userIndex, int relationType);)

Pre-Condition: N/A

Return:

- This function returns 1 if the message is related to the user.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Message is related to user	UserIndex = 1, senderID = 1	Return 1	Return 1	P
2	Message is not related to user	UserIndex = 1, senderID = 2	Return 0	Return 0	P
3	RelationType value is out-of-bounds	relationType = 5	Exit function	Exit function	P

checkInbox();

Function Description:

- This function prints the inbox messages of a specified user.

Prototype: checkInbox(user users, message msgs, int *msgCount, int userIndex, int userCount);

Pre-Condition: A message must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Function exits	Function exits	P
2	Valid user index is passed	users[userIndex].isDeleted = 0	Check inbox of user	Check inbox of user	P
3	User index with	users[userInd	Check inbox	Check inbox of	P

	incomplete data is passed	ex].name = '\0'	of user and skip incomplete data sections	user and skip incomplete data sections	

checkSent();

Function Description:
- This function prints the sent messages of a specified user.

Prototype: checkSent(user users, message msgs, int msgCount, int userIndex)

Pre-Condition: A message must exist.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Function exits	Function exits	P
2	Valid user index is passed	users[userIndex].isDeleted = 0	Check sent of user	Check sent of user	P
3	User index with incomplete data is passed	users[userIndex].name = '\0'	Check sent of user and skip incomplete data sections	Check inbox of sent and skip incomplete data sections	P

checkAnnouncements();

Function Description:
- This function prints all announcements.

Prototype: checkAnnouncements(user users, message msgs, int *msgCount, int userIndex, int userCount)

PROGRAM TEST SCRIPT, ROMAN

Pre-Condition: A message must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Function exits	Function exits	P
2	Valid user index is passed	users[userIndex].isDeleted = 0	Check sent of user	Check sent of user	P
3	User index with incomplete data is passed	users[userIndex].name = '\0'	Check announcements of user and skip incomplete data sections	Check announcements of user and skip incomplete data sections	P

modifyAccount();

Function Description:

- This function modifies a user's name & description

Prototype: modifyAccount(user users, int userIndex);

Pre-Condition: A user must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Account cannot be modified	Account cannot be modified	P

2	Valid user index is passed	users[userIndex].isDeleted = 0	Account cannot be modified	Account cannot be modified	P
3	User index with incomplete data is passed	users[userIndex].name = '\0'	Account still be modified	Account still be modified	P

modifySecurity();

Function Description:
- This function modifies a user's security answer & password.

Prototype: modifySecurity(user users, int userIndex)

Pre-Condition: A user must exist.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Deleted User index is passed	users[userIndex].isDeleted = 1	Account cannot be modified	Account cannot be modified	P
2	Valid user index is passed	users[userIndex].isDeleted = 0	Account cannot be modified	Account cannot be modified	P
3	User index with incomplete data is passed	users[userIndex].name = '\0'	Account still be modified	Account still be modified	P

connectionsModule();

Function Description:
- This function determines which connections function executes.

Prototype: connectionsModule(user users, int userIndex, int userCount)

PROGRAM TEST SCRIPT, ROMAN

Pre-Condition: A user must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Current user gets an announcement	tempID = 1, announcementCount =5	Announcement is visible	Announcement is visible	P
2	New user gets an announcement	tempID = 1, announcementCount =4	Announcement is visible	Announcement is visible	P
3					

addConnections();

Function Description:

- This function adds a personal connection via. selectUsers() or username.

Prototype: modifySecurity(user users, int userIndex)

Pre-Condition: A user must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Connection is self	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P
2	Connection is deleted	users[userCount].isDeleted = 1	Connection cannot be added	Connection cannot be added	P
3	Connection is a duplicate	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P

PROGRAM TEST SCRIPT, ROMAN

removeConnections();

Function Description:
- This function removes a personal connection via. selectUsers();

Prototype: removeConnections(user users, int userIndex, int userCount);

Pre-Condition: A user must exist.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	FFF	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P
2	Connection is deleted	users[userCount].isDeleted = 1	Connection cannot be added	Connection cannot be added	P
3	Connection is a duplicate	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P

viewPersonalConnections();

Function Description:
- This function views a user's personal connections.

Prototype: viewPersonalConnections(user users, int userIndex);

Pre-Condition: A user must exist.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail

1	Connection is self	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P
2	Connection is deleted	users[userCount].isDeleted = 1	Connection cannot be added	Connection cannot be added	P
3	Connection is a duplicate	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P

viewUserConnections();

Function Description:
- This function views all of a user's connections.

Prototype: viewUserConnections(user users, int userIndex, int userCount)

Pre-Condition: A user must exist.

Return:
- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Connection is self	userIndex = 1, tempID =1	Connection cannot be viewed	Connection cannot be viewed	P
2	Connection is deleted	users[userCount].isDeleted = 1	Connection cannot be added	Connection cannot be added	P
3	Connection is a duplicate	userIndex = 1, tempID =1	Connection cannot be added	Connection cannot be added	P

adminLogin();

Function Description:
- This function serves as an admin login interface.

PROGRAM TEST SCRIPT, ROMAN

Prototype: adminLogin(administrator admin, int *loggedIn);

Pre-Condition: An admin must exist.

Return:

- This function does not return anything as the parameters passed are changed through pointer; void.

#	Description	Input Data	Expected Output	Actual Output	Pass/Fail
1	Valid admin	admin= 1	Admin logs in	Admin logs in	P
3	No admin in the system	adminExits = 0	Login function exits	Login function exits	P