מגישים:

Michael ozeri 302444229

Ozeri.mich@gmail.com

Itai Shchorry 305584690

itaishch@gmail.com

Oded Navon 303066674

oded.navon@gmail.com

טלפון (זמין 24/7) : 0547939912 (מיכאל)

תיאור הפרוטוקול:

- מימשנו צד שרת בקובץ server.c וצד לקוח בקובץ client.c.
- פרוטוקול החיבור דרך sockets הוא פרוטוקול TCP כפי שנלמד בתרגולים / שיעורים.
- נדרש להריץ את צד השרת לפני הרצת צד הלקוח (כדי שמישהו יאזין בצד השני)
- המימוש:
  - החזקה של מערך של struct (נקרא *userDB) שבכל מיקום שלו מכיל מצביע לstruct מסוג:

```
typedef struct USERDB
{
    char name[MAX_USERNAME];
    char password[MAX_PASSWORD];
    int numOfEmails;
    email emails[MAXMAILS];
    int isActivated;
    int userIndexInClientHandler;
} userDB;
```

  - בעצם לכל client שקיים במערכת יש שם, סיסמה, מספר אימיילים שיש לו בinbox, ואת מערך הemail ששומר את כל הinbox שלו. מימוש כל email בצורה הבאה באמצעות struct:

```
typedef struct EMAIL
{
    //mail id is the place in the array of the mail
    char from[MAX_CONTENT];
    char to[MAX_CONTENT];
    char subject[MAX_CONTENT];
    char content[MAX_CONTENT];
    bool deleted;
} email;
```

  - באתחול השרת בעצם רץ על קובץ הdatabase ומאתחל userDB לכל אחד מהמשתמשים
  - כמו כן לכל משתמש יש שדה המציין האם הוא מחובר כרגע, ושדה אינדקס המציין אינדקס במערך מהstruct הבא, השומר נתונים על כל המשתמשים המחוברים:

```
typedef struct clientHandler_t{
    int fd;
    int confirmedUser;
    int isActivated;
    char* username;
    int userIndexInDB;
} clientHandler;
```

- בעצם מבנה הנתונים יהיה מערך של הstruct הנ"ל השומר את השדות: (לכל משתמש)
  - הfile descriptor של המשתמש המחובר / מתחבר
  - האם המשתמש כבר אימת את הזהות או רק עשה connect? (confirmedUser)
  - שם המשתמש שלו (יוזן לאחר האימות)
  - האינדקס של המשתמש המחובר בUserDB.(מסד הנתונים של כל היוזרים המחוברים)
  - isActivated – האם היוזר כרגע מחובר ו online.
- כל שורה שהמשתמש מכניס בכל פונקציונליות נכנסת כשורה נפרדת שלאחריה לוחצים enter.
- פונקציונליות:
  - **QUIT –** בעצם ניתוק של הsocket הנוכחי שהשרת נמצא אתו בתקשורת, מצד הלקוח יש התנתקות מהשרת ולא ניתן יותר להריץ פקודות עד לחיבור מחדש, מצד השרת יש חזרה ללולאת while שמאזינה לחיבור של לקוח חדש. (השרת לא "יוצא / נכבה" לבד)
  - אתחול מערך ה*userDB נמצא מחוץ ללולאת הwhile שמחכה כל פעם לclient חדש ולכן אם נשלחו מיילים מסוימים הם שמורים עבור כל client מהdatabase בנפרד אצלו בuserDB.
  - **SHOW_INBOX** – השרת רץ על כל המיילים של המשתמש בתוך המערך email שבstruct שלו ובודק האם הemail.deleted == true. בעצם מחיקה של מייל מומשה באמצעות סימון המייל כ"נמחק" – כלומר לא באמת נמחק, יכול לספק הכנה להרחבה עתידית של מציאת מיילים מחוקים. (המימוש הזה אושר לנו במייל מהמתרגל...)
  - **GET_MAIL mail_id** – הלקוח שולח דרישה למייל מס' ... – מספר המייל ממומש ע"י מיקומו במערך הemail בתוך הstruct של הuserDB הספציפי שאנחנו מחוברים אליו. ולכן הפונקציה תדפיס את הemail שנמצא במיקום הmail_id במערך.
  - **DELETE_MAIL mail_id –** באותה צורה השרת הולך למיקום הemail[mail_id] במערך ומסמן את השדה של הstruct email שם להיות deleted = true
  - **COMPOSE :**
    - כל ההודעה נשמרת ב string אחד מהצורה: COMPOSE To:<user>\nSubject:<subject>\nTex:<text>
    - ההודעה נשלחת בצורה הזאת לשרת שמפרסר אותה ומכניס למסד נתונים
    - הפונקציה בצד השרת רצה על הנתונים ב*userDB ומשווה את שם הנמענים לשמות במערך – אם הם שווים, המייל החדש נשמר אצלם בstruct.
    - השרת ישלח חזרה mail sent ללקוח.
    - הערה: במידה ויש נמען בשורת הto אשר לא נמצא כuser בשרת שלנו אנחנו נתעלם ממנו ופשוט לא נשלח לו את הmail. (גם אם כל הנמענים לא נמצאים)
    - הערה: אם שלחנו to: Oded, Oded, Oded המייל ישלח פעם אחת לOded.
  - **SHOW_ONLINE_USERS**:
    - מציג את כל המשתמשים המחוברים כרגע לפי שדה הisActivated במערך הfdArray.

- **MSG** - לאחר השימוש בפונקציה הזו יוכל המשתמש לשלוח הודעה למשתמש אחר שמחובר באמצעות MSG itai: hello world.
- במידה ובין ההצגה של SHOW_ONLINE_USERS לבין שליחת הודעת הצאט התנתק הנמען ישלח מייל בפורמט שנתבקש בתרגיל הבית.

- **פונקציות השליחה / קבלה:**
  - שוות בשני הצדדים.
  - כוללות פונקציית שליחה int sendFullMsg(int sock, char* buf) שעוטפת את הפונקציות:
    - sendLenOfMsg – שליחת אורך ההודעה לפני השליחה של הטקסט עצמו
    - sendAll – שליחת ההודעה עצמה.
  - כנ"ל פונקציית מעטפת recvFullMsg שעוטפת את הפונקציות:
    - recvLenOfMsg – קבלת אורך ההודעה הצפויה להישלח
    - Recvall – קבלת ההודעה עצמה.
  - פונקציות המעטפת הן אלו שנקראות למעשה בשני הצדדים על מנת לוודא את קבלת ההודעה כמו שצריך, הן מחשבות את אורך ההודעה שצריך להישלח ושולחות אותו לפני ההודעה בint כדי שפונקציית הקבלה תדע מה אורך ההודעה שהיא צריכה לקבל.
- הערות נוספות / מקרי קצה:
  - אם הclient מכניס שם משתמש או סיסמה לא נכונים – הוא מתנתק מהשרת. השרת לא מפסיק את עבודתו.
  - במידה ויש נמען בשורת הto אשר לא נמצא כuser בשרת שלנו אנחנו נתעלם ממנו ופשוט לא נשלח לו את הmail. (גם אם כל הנמענים לא נמצאים)
  - אם שלחנו to: Oded, Oded, Oded המייל ישלח פעם אחת לOded.
- העדכון המרכזי מהתרגיל הקודם:
  - 1 הוספת מבנה נתונים בצד השרת על מנת לתמוך בשירותי online. (מערך של clienthandler)
  - השרת עובד בשיטת דגימה באמצעות לולאת while וselect שנמצא בתוכה.
  - לפני כל איטרציה טוענים את כל הfd הרלוונטים ממערך הfdarray לset שלנו ובודקים מי רוצה לשלוח הודעה.
    - אם בשרת החיבור יש בקשה לaccept אנחנו מייצרים את החיבור החדש ומוסיפים לfdarray
    - אם יש משתמש מחובר כבר שרוצה לשלוח הודעה אנחנו מקבלים את ההודעה שלו ומטפלים בה באמצעות הפונקציה HandleClientMessage.
    - באותה צורה עובד צד הלקוח כאשר הכנסנו לselect את הstdin על מנת לתמוך בשירותי צאט (כמובן בכל איטרציה של הwhile) וגם את הsocket של השרת שבאמצעותו הם מתקשרים.
  -

- מבנה כללי של "תוכנת" הserver.c להבנה בhigh level:

```
int main(int argc, char** argv)
{

   //create and parse user database

    //listen for a new connection

    //create array fdArray

    //create fd_set

    while(1)
    {
        //insert all valid file descriptors into fd_set


        select();


        //if client is trying to connect - accept it

       //if a connected user is trying to send a messege take care of it using handler functions
    }
    free(userdb); //TODO - how we release this?!
    close(serverSocket); //TODO - how we release this?!
    return 0;
}
```

בדיקה נעימה!