רשתות תקשורת ומחשבים תרגיל 1

מגישים:

- שם: דניס גוליאנוב, ת.ז. 320882988, אימייל: denisg@mail.tau.ac.il
- שם: דוד ליפשיץ, ת.ז. 206107740 אימייל: davidl2@mail.tau.ac.il

תיאור הפרוטוקול:

1. לאחר שה-client וה-server יצרו ביניהם חיבור (מעל TCP) השרת שולח **הודעת פתיחה** עם המבנה:

0 or 1 (byte)	Heap 0 size (short)	Heap 1 size (short)	Heap 2 size (short)	Heap 3 size (short)

כאשר ה-byte הראשון מכיל את הערך isMisere (0 עבור משחק רגיל, 1 עבור משחק misere)

לאחר מכן, יופיעו 4 shorts שמתארים את גדלי הערימות, כאשר Heap 0 הוא ה-heap השמאלי ביותר (heap A)

2. בכל תור של המשתמש, לאחר שה-client קלט הנתונים מהמשתמש: מספר ערימה וכמה איברים להסיר ממנה, ה-client ישלח בקשה לשרת לבצע מהלך. להלן מבנה ההודעה (client query):

0,1,2 or 3 (byte)	Num items to remove from heap (short)

כאשר ה-byte הראשון מכיל את מספר הערימה ממנה המשתמש רוצה להסיר איברים (הערימה השמאלית ביותר, ערימה A, היא בעלת מספר סידורי 0). לאחר מכן, יופיע short אחד שמכיל את מספר האיברים שיש להסיר מהערימה עם המספר הסידורי שמופיע ב-byte ההתחלתי.

3. בכל תור של ה-client, השרת קולט את הבקשה שתוארה בסעיף 2 (client query), מבצע את הצעד המבוקש (אם חוקי), מבצע צעד משלו (לפי הלוגיקה שהוגדרה עבור השרת) ומחזיר תשובה ל-client:

Flag container byte	Heap 0 size (short)	Heap 1 size (short)	Heap 2 size (short)	Heap 3 size (short)

כאשר ה-byte הראשון (flag container byte) מכיל דגלים על מצב משחק. המבנה שלו הוא:

*****CBA

A– (least significant bit): יקבל 1 אמ"מ המהלך האחרון של ה-client (client query) היה חוקי.

B– יקבל 1 אמ"מ המשחק הסתיים

C– במקרה ש B קיבל את הערך 1 (דהיינו, המשחק הסתיים), C יקבל 1 אם השרת ניצח ויקבל 0 אם הclient ניצח.

לאחר ה-flag container byte יופיעו 4 shorts עם מספרי האיברים בכל ערימה. (כמו מקודם, heap 0 מכיל את מספר האיברים בערימה השמאלית ביותר, ערימה A)

הערה על הפרוטוקול: גדלי ה-heaps בכל הודעה, מיוצגים על ידי short unsigned.

אופן הרצת התוכנית

מקמפלים את הקבצים על VM בעזרת ה-make file (פקודת make all). קובץ ההרצה nim מתאים ללקוח וקובץ ההרצה nim_server מתאים לצד שרת.

מבנה התוכניות

nim_protocol_tools.c, nim_protocol_tools.h– מגדירים קבועים ומתודות הקשורים לפרוטוקול שתואר מעלה, לצורך שימוש על ידי השרת והלקוח. למשל CLIENT_QUERY_SIZE מתאר את גודל ההודעה ששולח הלקוח לשרת בבתים. כמו כן, הקבצים מספקים שירותים לטיפול ב-flag container byte כפי שתואר בפרוטוקול. כך למשל, setGameOver(unsigned char* container) מדליקה את הביט B שמשמעותו כי המשחק הסתיים. דוגמה נוספת: lastMessageAcked(unsigned char container) מחזירה ערך חיובי אם הדגל A (אישור מהלך לקוח) דולק ב-container, אחרת מחזירה 0.

client.c, client.h- מימוש צד לקוח. הקובץ אחראי על קריאת הפרמטרים של הקלט, יצירת חיבור עם צד השרת, קריאת קלט מהמשתמש (לצורך ביצוע מהלך) וניהול משחק nim עם השרת לפי הפרוטוקול שהוגדר.

client_game_tools.c, client_game_tools.h-מתודות עזר עבור client.c. מטפלות בפלט למשתמש: הדפסת כותרות, בקשות, הדפסת זהות המנצח, הדפסת אישורי מהלכים והדפסת גדלי הערימות כפי שהוגדר.

socket_IO_tools.c, socket_IO_tools.h- מתודות שימושיות לשימוש השרת והלקוח הקשורות ב-sockets, כגון: סגירת socket עם בקרת שגיאה, שליחה של מידע (send_all) וקבלת מידע (recv_all) דרך socket.

nim_game.c, nim_game.h- מימוש של המשחק עצמו עבור צד שרת. המחלקה מגדירה ושומרת על קבועים של המשחק (כמו גדלי הערימות וסוג המשחק- isMisere). לכל מי שעושה include לnim_game.h יש גישה למשתנים אלו. פונקציות עיקריות המוגדרות בקובץ:

init_game- פונקציה זו מקבלת כפרמטרים את סוג המשחק וגודל ההתחלתי של הערימות ומאתחלת משחק חדש.

makeRound- פונקציה זו מקבלת כפרמטרים את המהלך של הלקוח (מספר ערמה וכמות פריטים לקחת), מבצעת את המהלך של הלקוח ואת הלוגיקה של השרת ומחזירה האם המהלך היה חוקי וערך (int) האומר האם המשחק נגמר ואם כן מי ניצח בו.

game_server.h,**game_server.c** - מימוש צד השרת. הקובץ אחראי על קריאת הפרמטרים של הקלט, יצירת חיבור עם צד הלקוח וניהול התקשורת מולו לפי הפרוטוקול (יצירה ושליחה של הודעות, קבלת הודעות מהלקוח ופענוחן ע"פ הפרוטוקול). מחלקה זו משתמשת במחלקה nim_game לצורך ניהול המשחק ונעזרת בפונקציות של socket_IO_tools.

הערה: תיעוד מתודות ניתן למצוא בקבצי המקור עצמם.

טיפול בשגיאות ומקרי קצה

<u>ה-client:</u>

- קלט מהמשתמש **לצורך ביצוע מהלך** צריך להיות מהצורה (כמובן ללא הסוגריים המרובעים):
  [A-D] [SPACE] [**unsigned** short]
  (כאשר המספר האחרון הוא חיובי ממש, SPACE הוא תו רווח אחד או יותר) למשל: A 500
  או מהצורה
  [A-D] [**unsigned** short]
  (כלומר כמו מקודם אך ללא הרווח) למשל: D29
  קלט שהוא בקשה **לסגירת התוכנית** צריך להיות התו Q, או כל מחרוזת אחרת שמתחילה ב-Q (לא נקרא את יתר התווים אחרי שקראנו Q) למשל: Q או QAFD או Q543294.

  קלט שלא עונה על הפורמט שתואר מעלה יחשב כקלט לא תקין לתוכנית, התוכנית תדפיס הודעת שגיאה מתאימה ותסיים את ריצתה.

- **אם הלקוח מחכה לקלט מהמשתמש ובזמן זה השרת נסגר, אז הלקוח יודיע על סגירת הקשר רק לאחר שהמשתמש הזין את הקלט. במקרה שבו הנתונים שנקלטים מהמשתמש לא חוקיים (דהיינו, לא ענו על הפורמט שתואר בסעיף הקודם) תודפס הודעת שגיאה על קלט לא תקין ולאחר מכן התוכנית תסתיים מבלי להדפיס כי השרת סגר את הקשר!**
- במקרים הבאים:
  1. קלט בשורת הפקודה לתוכנית אינו תקין (לא כפי שהוגדר בתרגיל)
  2. שגיאה במציאת כתובת IP של host (שירות DNS) או שגיאה בהתחברות אל השרת
  3. שגיאה ביצירת או סגירת שקע (socket)
  4. שגיאה בקבלת מידע מהשרת, שגיאה בשליחת מידע לשרת

  תודפס שגיאה מתאימה (לרבות יחד עם strerror(errno)), המשאבים יפונו והתוכנית תסתיים.
- משום שבלקוח אנו משתמשים במתודה getaddrinfo, ניתן להכניס בפרמטר של ה-port גם שם של השירות המתאים לפורט, למשל "http" (המתאים לפורט 80).

<u>ה-server :</u>

במקרים הבאים:

1. קלט לתוכנית אינו תקין (לא כפי שהוגדר בתרגיל)

2. שגיאה ביצירת או סגירת שקע (socket)

3. שגיאה בbind לפורט הרצוי

4. שגיאה בהקשבה על הport

5. שגיאה באישור בקשה מהלקוח (accept).

3. שגיאה בקבלת מידע מהשרת, שגיאה בשליחת מידע ללקוח

4. סגירה לא צפויה של החיבור מצד הלקוח (לא צפויה לפי הפרוטוקול)

במקרים אלה תודפס הודעת שגיאה המתארת את המצב, המשאבים יפונו והתוכנית תסתיים.

הערה: כאשר השרת מקבל בקשה למהלך מהלקוח, אם מספר הערימה המצוין בבקשה גדול מדי (דהיינו, גדול מ-3) אז המהלך יחשב על ידי השרת כמהלך לא חוקי והריצה תמשיך (השרת לא יאשר את המהלך). ה-client במקרה שלנו ישלח רק מספרי ערימות חוקיים, אך הוספנו בדיקה זו בכל מקרה.

מידע חשוב נוסף

1. במקרה שבו שולחים מידע לצד השני והצד השני סוגר את החיבור, לא יופעל SIGPIPE ולכן לא תמצא מימוש ל-handler משלנו שתופס את הסיגנל. מאחר ואנחנו שולחים מידע באמצעות המתודה:

```
int send_all(int sockfd, char* buffer, int num_bytes, int* connection_closed)
```

(שנמצאת ב- socket_IO_tools.c) תמיד שליחה מתבצעת עם הדגל `MSG_NOSIGNAL`, שמבטל שליחת סיגנל לתהליך. במקום זאת, המתודה send_all מבררת לפי errno האם מדובר ב-EPIPE (סגירת הקשר מצד שני), אם כן, connection_closed* יקבל את הערך 1 (ואחרת 0).

2. ב-make file מופיע הדגל -std=gnu99 לצורך שימוש במתודות getaddrinfo וכן `gai_strerror`.