# תרגיל בית 4 – אבן נייר, מספריים, לטאה, ספוק

**מועד הגשת התרגיל: עד יום שבת 25/01/20 בשעה 23:55. לא תהינה דחיות**

*בודק אחראי: אורי ברכה uribracha@mail.tau.ac.il*

## מטרת התרגיל

- העמקת ההבנה במושגי החוט (Thread) במערכות הפעלה בכלל וב-Windows בפרט.
- עבודה עם מספר חוטים במקביל.
- שימוש ב-Mutex וב-Semaphore לסיכנרון גישה למשאבים משותפים בין חוטים.
- שימוש ב-Mutex ו-Semaphore לסנכרון גישה לזיכרון משותף בין החוטים.
- העמקת ההבנה בתקשורת מחשבים.
- שימוש ב-TCP Sockets:
  - WSAStartup
  - WSACleanup
  - socket
  - bind
  - listen
  - accept
  - recv
  - connect
  - closesocket
  - send

## הגשה

צורת ההגשה מפורטת במסמך "הנחיות להגשת תרגילי בית – תש"ף" שבאתר המודל. אנה הקפידו למלא אחר ההוראות.

בנוסף, אנא הקפידו על ההנחיות הבאות.

- הגישו פרוייקט מלא, כולל קבצי פרויקט (*.sln, *.vcxproj, *.vcxproj.filters) של Visual Studio 2017, באופן שיאפשר לבודק התרגילים לפתוח את הפרויקט על ידי לחיצה כפולה על קובץ ה-solution ולקמפל את הפרויקט ללא התראות או שגיאות.
- הגישו בנוסף את תיקיית ה-Debug עם קובץ ה-exe.
- בפרויקט הזה תגישו שני פרויקטים בתוך solution אחד. שם ה-solution צריך להיות groupXX_ex4. שמות הפרויקטים צריכים להיות groupXX_ex4_client ו-groupXX_ex4_server. שמות ה-executables צריכים להיות groupXX_ex4_client.exe - groupXX_ex4_server.exeבהתאמה. החליפו את XX בשתי הספרות של מספר הקבוצה שלכם.
- שם ה-zip שאתם מגישים צריך להיות group_XX_ex4.zip.
- שימו את הקבצים המשותפים לשני הפרויקטים בתיקייה בשם Share, שממוקמת ביחד עם תיקיות הפרויקט בתיקיית ה-solution.
- קמפלו את הקוד לגרסאת 64-bit.
- הקוד לא צריך לתמוך ב-Unicode. וודאו בהגדרות הפרויקט, שאתם לא מקמפלים ל-UNICODE: בהגדרות הפרויקט תחת General בשורה Character Set יש לבחור-Use Multi Byte Character Setולא Use Unicode Character Set. לאחר קביעת ההגדרה הזאת, אפשר להתייחס ל-TCHAR כמו ל-char.

## דגשים

- מענה לשאלות בפורום – היות ומדובר בתרגיל ארוך שיתפרס מפרסומו עד סוף הסמסטר המענה בפורום ישתנה לאורך הזמן. קראו בקפידה את התרגיל בשבוע הראשון מפרסומו ותכננו את הקוד שלכם – חשבו מה אתם לא מבינים. בשבוע הראשון לא יהיה מענה יומיומי בפורום. במהלך השבוע השני מפרסום התרגיל יינתן מענה יומיומי לשאלות בפורום – נצלו את השבוע השני לשאול את כל השאלות שיש לכם. בשבועות השלישי והרביעי מפרסום התרגיל המענה בפורום לא יהיה יומיומי אך ייעשה מאמץ לענות אחת לכמה ימים.
- הקפידו על קוד קריא ומתועד בצורה ממוקדת ועניינית. אם אתם לא בודקים משהו – הסבירו מדוע. אם הטיפול שלכם הוא ייחודי – הסבירו. בפרט הקפידו על חלוקה למודולים, פונקציות ומתן שמות משמעותיים לפונקציות ומשתנים והימנעו מקבועים בקוד (השתמשו ב-define).
- רשמו לעצמכם את מבנה התוכנה הכללי לפני שאתם מתחילים לקודד.
  - חשבו איזה מודולים ופונקציות אתם צריכים. מתוך הפונקציות, איזה יהיו סטטיות ואיזה פומביות. אל תכתבו את כל התוכנה בקובץ אחד!
  - זכרו כי כל קטע קוד שאתם משתמשים בו יותר מפעם אחת, צריך להיכתב כפונקציה נפרדת. כאשר פונקציה נעשית גדולה ומסובכת, פצלו אותה למספר פונקציות.
- זכרו להשתמש בכלי הדיבוג שה-IDE מספק.
- איטרציות – שימו לב שאתם מקדמים את שתי התכניות במקביל כדי שתוכלו לבדוק את עצמכם ולזהות בעיות בתקשורת ביניהם כמה שיותר מוקדם.
- אתם יכולים לדבאג שני תהליכים, אחרי שהרצתם את אחת התכניות ב-Debug ב- Visual Studio תוכלו לדבאג במקביל גם את התכנית השנייה לאחר שהיא התחילה לרוץ: Debug -> Attach to process ולבחור את התהליך.
- באג ב-Design הוא אתם יודעים מה ב-Debug – תכנון טוב יחסוך לכם הרבה עבודה.
  - חשבו איזה מודולים ופונקציות אתם צריכים. מתוך הפונקציות, איזה יהיו סטטיות ואיזה פומביות. אל תכתבו את כל התוכנה בקובץ אחד!
  - זכרו כי כל קטע קוד שאתם משתמשים בו יותר מפעם אחת, צריך להיכתב כפונקציה נפרדת. כאשר פונקציה נעשית גדולה ומסובכת, פצלו אותה למספר פונקציות.
- אתחלו את כל המשתנים שלכם לערך של שגיאה (מצביעים ל-NULL, אינדקס ל-1- וכו').
- השתמשו בזיכרון דינמי לאחסן מידע שגודלו אינו ידוע בזמן הקומפילציה. אינכם רשאים להניח
חסם עליון שרירותי לגודל המידע. השתמשו בקבועים ושימו לב לשחרור זיכרון דינמי.
- אתחלו את כל הפוינטרים ל-NULL. כל פונקציה שמקבלת מצביע צריכה לבדוק שהוא שונה מ-NULL לפני שהיא עושה dereference (אופרטור *).
- בדקו את ערך החזרה של כל פונקציה, שיכולה להחזיר שגיאה (malloc, WaitForSingleObject וכו'). פעלו בהתאם לערך.
- שחררו זיכרון דינאמי ו-handles בהקדם האפשרי (באמצעות Free ו-CloseHandle בהתאמה).
- לפני שאתם משתמשים בפונקציית WinAPI, רצוי לקרוא את התיעוד שלה ב-MSDN שלה. באופן כללי, רצוי גם לקרוא את הפונקציות שמופיעות ב-MSDN תחת Related Functions.

## הנחיות והנחות

- **חובה** לבדוק את **כל** ערכי ההחזרה הרלוונטיים (אם יש ספק, תבדקו הכל) של **כל** הפונקציות שאתם משתמשים.
- **חובה** לשחרר משאבים בכל תרחיש. משאבים הם זיכרון שהוקצה דינאמית, HANDLE-ים של מערכת ההפעלה (סוקטים, קבצים, מיוטקסים וכו').
- **אין** להשתמש ב-exit כדי לסיים את ריצת התכנית. הדרך היחידה שבה התכנית תסיים את ריצתה היא דרך return 0 ב-main שלה.
- אין להשתמש ב-TerminateProcess.
- השימוש ב-TerminateThread מותר אחר ורק לאחר ניסיון סגירה שנכשל בגלל Timeout.
- **בכל** מקרה של שגיאה יש להדפיס הודעת שגיאה **ייחודית** לאותה השגיאה שמפרטטת מה קרה במלל ולסיים את ריצת התכנית (כאמור, דרך ה-return ב-main ולא באמצעות exit).
- **מומלץ** מאוד להשתמש ב-lblCleanup (חפשו בגוגל או קראו בתחתית מפתח הניקוד של תרגיל בית 1).
- **זכרו** – גם אם בתרגול לא ראיתם את משהו אבל הוא כתוב בתיעוד המלא של הפונקציה – אתם אמורים לדעת את זה. לא משתמשים בפונקציה בלי לקרוא את התיעוד שלה באינטרנט.
- **אין** לכתוב פונקציות ארוכות – לכל היותר 100 שורות. מצד שני אל תכתבו פונקציה לכל 3 שורות תוודאו שהחלוקה לפונקציות היא מונחית מטרה, ממזערת שכפול קוד ו/או משפרת קריאות.
- **יש** לתת שמות משמעותיים לפונקציות ומשתנים וכן לבצע חלוקה למודולים.

**טיפ של אלופים**: מומלץ **מאוד** לעבוד עם שירות Version Control כלשהו למשל כמו GitHub על מנת לשמור את ההתקדמות שלכם וזאת משום שהתרגיל הוא איטרטיבי. ביצוע commit לאחר השלמת מדרגה בתרגיל יעזור לכם לעקוב אחר ההתקדמות שלכם, לזהות בעיות ויאפשר לכם לחזור לנקודה האחרונה שבה הקוד שלכם עבד במקרה שהסתבכתם ואתם לא מצליחים למצוא את הבעיה. למי מכם שמעולם לא השתמש ב-Git ו-GitHub עדיף מאוחר מאשר אף פעם.

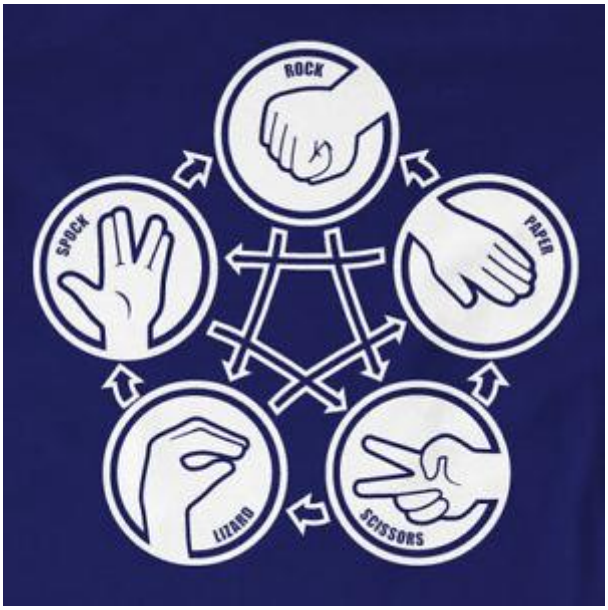
בהצלחה!

## סקירה כללית

בתרגיל זה תממשו גרסה מקוונת של המשחק אבן נייר, מספריים, לטאה, ספוק.

חוקי המשחק דומים מאוד לאבן נייר ומספריים כפי שניתן לראות בהדגמה כאן.

לנוחיותכם מצורפת הדיאגרמה הבאה שמכסה את כל השילובים האפשריים במשחק כאשר חץ מסמן ניצחון. לדוגמה, לטאה מנצחת נייר ואבן מנצחת לטאה.




עלכים לכתוב תכנית שרת ותכנית לקוח.

עליכם לממש שתי תוכנות (וכפועל יוצא של כך יהיו לכם שני פרויקטים ב-Solution):

1. תוכנת שרת שמקבלת תקשורת נכנסת מתכנת הלקוח, מבצעת את הלוגיקה של המשחק ואת שאר הפעולות שיפורטו בהמשך ומחזירה תקשורת לתוכנות הלקוח המחוברות אליה. תוכנת השרת צריכה להיות רובסטית ועצמאית גם בהיעדר לקוחות שמנסים להתחבר אליה ולאפשר סגירה שלה בצורה אלגנטית על ידי מי שהריץ אותה. בנוסף על השרת לתחזק קובץ של לוח אליפויות בשם Leaderboard.csv (יוסבר בהמשך).
2. תוכנת לקוח שמהווה את הממשק של המשתמש במשחק ומתחברת לתוכנת השרת. דרך תוכנת הלקוח המשתמש יכול לשחק ולעשות פעולות נוספות שיפורטו בהמשך. תוכנת הלקוח צריכה להיות רובסטית ועצמאית גם אם אין שרת להתחבר אליו ולאפשר למשתמש לנסות להתחבר שוב או לצאת.

**הערות**:

- סדר הפעלת התוכנות לא אמור להשפיע על פעילותן התקינה. כלומר, אפשר להריץ את השרת לפני שיש לקוחות ואחרי שיש לקוחות ואפשר להריץ לקוחות כשאין שרת וכשיש שרת ובכל מקרה שבו יש שרת ולקוח ללא תלות בסדר ההרצה שלהם הלקוח יצליח (אלא אם יש שגיאה אחרת) להתחבר לשרת.
- בכל מקום בתרגיל שבו תכנת הלקוח מבקשת קלט מהמשתמש (למעט בשורת הפקודה) לאחר הצגת תפריט עם אפשרויות המשתמש יבחר מהתפריט אופציה על ידי הקלדת המספר שלה בתפריט.

# ממשק שורת פקודה (Command Line Interface)

## תוכנת השרת

קריאה לתכנת השרת נראית כך:

Server.exe <port>

- port – מספר ה-port שדרכו השרת יאזין ללוקחות שמנסים להתחבר.

לדוגמה:

```
>C:\...\groupXX_ex4_server.exe 8888
```

## תוכנת הלקוח

קריאה לתוכנת הלקוח נראית כך:

Client.exe <server ip> <server port> <username>

- server ip – כתובת ip של השרת
- server port – כתובת port של תוכנת השרת
- username – שם המשתמש (שימו לב שאין חשיבות לאותיות גדולות או קטנות).

שימו לב שיש רווחים בין הפרמטרים.

לדוגמה:

```
>C:\...\groupXX ex4 client.exe 127.0.0.1 8888 Sheldon
```

ניתן להניח ששם המשתמש הוא ייחודי (כלומר אף פעם לא יתחברו שני לקוחות עם אותו שם משתמש) ומכיל אך ורק אותיות ומספרים ללא רווחים. כמו כן ניתן להניח שאורך שם המשתמש יהיה לכל היותר 20 תווים.

# Happy Flow

להלן דוגמאות לריצות אידיאליות של שתי התוכנות במקרה שאין בו תקלות.
למען פשטות, הושמטו פרטי ממשק המשתמש (קלט פלט למסך ולקבצים) מהטבלה.

| | מבצע | פעולה | סוג הודעה |
|---|---|---|---|
| 1 | שרת | מחכה ל-connection. | |
| 2 | לקוח 1 | מתחבר לשרת ושולח את שם המשתמש לשרת | CLIENT_ REQUEST |
| 3 | שרת | מאשר את הלקוח. | SERVER_ APPROVED |
| 4 | שרת | מציג לו את האפשרויות הבאות:<br>1. לשחק נגד שחקן אחר (אם מחובר)<br>2. לשחק מול "המחשב"<br>3. לצפות בלוח האליפויות<br>4. להתנתק מהשרת | SERVER _MAIN_ MENU |

מכאן הריצה מתפצלת בהתאם למה שהמשתמש יבחר.

1. לשחק נגד שחקן אחר

| | מבצע | פעולה | סוג הודעה |
|---|---|---|---|
| 5 | לקוח 2 | מתחבר לשרת ושולח לו את שם המשתמש | CLIENT_ REQUEST |
| 6 | שרת | מאשר את לקוח 2. | SERVER_ APPROVED |
| 7 | לקוח 1 | בוחר לשחק נגד שחקן אחר | CLIENT_VERSUS |
| 8 | לקוח 2 | בוחר לשחק נגד שחקן אחר | CLIENT_VERSUS |
| 9 | שרת | שולח ללקוח 1 וללקוח 2 שעומד להתחיל משחק | SERVER_INVITE |
| 10 | שרת | מבקש מלקוח 1 ומלקוח 2 לבחור עם מה לשחק מתוך אבן, נייר, מספריים, לטאה או ספוק | SERVER_PLAYER_MOVE_REQUEST |
| 11 | לקוח 1 | בוחר מהרשימה מהלך ושולח לשרת | CLIENT_PLAYER_MOVE |
| 12 | לקוח 2 | | |
| 13 | שרת | משווה בין המהלכים שבחרו לקוח 1 ולקוח 2 ומחזיר להם את תוצאת המשחק | SERVER_GAME_RESULTS |
| 14 | שרת | מציע ללקוח 1 ולקוח 2 את האופציות הבאות:<br>1. לשחק שוב זה נגד זה.<br>2. לסיים את המשחק ולחזור לתפריט הראשי. | SERVER_GAME_OVER_ MENU |
| 15 | לקוח 1 | בוחר לשחק שוב | CLIENT_REPLAY |
| 16 | לקוח 2 | | |
| 17 | שרת | מבקש מלקוח 1 ומלקוח 2 לבחור עם מה לשחק מתוך אבן, נייר, מספריים, לטאה או ספוק | SERVER_PLAYER_MOVE_REQUEST |
| 18 | לקוח 1 | בוחר מהרשימה מהלך ושולח לשרת | CLIENT_PLAYER_MOVE |
| 19 | לקוח 2 | | |
| 20 | שרת | משווה בין המהלכים שבחרו לקוח 1 ולקוח 2 ומחזיר להם את תוצאת המשחק | SERVER_GAME_RESULTS |
| 21 | שרת | מציע ללקוח 1 ולקוח 2 את האופציות הבאות:<br>1. לשחק שוב זה נגד זה.<br>2. לסיים את המשחק ולחזור לתפריט הראשי. | SERVER_GAME_OVER_ MENU |
| 22 | לקוח 1 | בוחר לשחק שוב | CLIENT_REPLAY |
| 23 | לקוח 2 | בוחר לא לשחק שוב | CLIENT_MAIN_MENU |
| 24 | שרת | מיידע את לקוח 1 שלקוח 2 לא רצה לשחק שוב | SERVER_OPPONENT_QUIT |
| 25 | שרת | מחזיר את לקוח 1 ולקוח 2 לתפריט הראשי:<br>1. לשחק נגד שחקן אחר (אם מחובר)<br>2. לשחק מול השרת<br>3. לצפות בלוח האליפויות<br>4. להתנתק מהשרת | SERVER_MAIN_MENU |
| 26 | לקוח 1 | בוחר לשחק נגד שחקן אחר | CLIENT_VERSUS |
| 27 | לקוח 2 | בוחר להתנתק מהשרת | CLIENT_DISCONNECT |

| | | | |
|---|---|---|---|
| 28 | שרת | אין שחקנים אחרים לשחק מולם אז שולח ללקוח 1 שהוא לא הצליח למצוא מול מי לשחק | SERVER_NO_OPPONENTS |
| 29 | שרת | מחזיר את לקוח 1 ולקוח 2 לתפריט הראשי:<br>1. לשחק נגד שחקן אחר (אם מחובר)<br>2. לשחק מול השרת<br>3. לצפות בלוח האליפויות<br>4. להתנתק מהשרת | SERVER_MAIN_MENU |
| 30 | לקוח 1 | בוחר להתנתק מהשרת | CLIENT_DISCONNECT |
| 31 | שרת | השרת ממשיך לרוץ עד שמי שהריץ אותו מכבה אותו | |

## 2. לשחק מול "המחשב"

| | מבצע | פעולה | סוג הודעה |
|---|---|---|---|
| 5 | לקוח 1 | בוחר לשחק מול השרת | CLIENT_CPU |
| 6 | שרת | מגריל באקראי מהלך מבין אבן, נייר, מספרים, לטאה או ספוק | |
| 7 | שרת | שולח ללקוח בקשה לקלט | SERVER_PLAYER_MOVE_REQUEST |
| 8 | לקוח 1 | בוחר מבין האופציות: אבן, נייר, מספרים, לטאה או ספוק ושולח לשרת | CLIENT_PLAYER_MOVE |
| 9 | שרת | מחשב את תוצאות המשחק | SERVER_GAME_RESULTS |
| 10 | שרת | מציע ללקוח 1 את האופציות הבאות:<br>1. לשחק שוב זה נגד זה.<br>2. לסיים את המשחק ולחזור לתפריט הראשי. | SERVER_GAME_OVER_ MENU |
| 11 | לקוח 1 | בוחר שלא לשחק שוב | CLIENT_MAIN_MENU |
| 12 | שרת | מציג לו את האפשרויות הבאות:<br>5. לשחק נגד שחקן אחר (אם מחובר)<br>6. לשחק מול "המחשב"<br>7. לצפות בלוח האליפויות<br>8. להתנתק מהשרת | SERVER _MAIN_ MENU |
| 13 | לקוח 1 | בוחר להתנתק מהשרת | CLIENT_DISCONNECT |
| 14 | שרת | השרת ממשיך לרוץ עד שמי שהריץ אותו מכבה אותו | |

## 3. לצפות ב-Leaderboard

| | מבצע | פעולה | סוג הודעה |
|---|---|---|---|
| 5 | לקוח 1 | בוחר לצפות בלוח האליפויות | CLIENT_LEADERBOARD |
| 6 | שרת | שולח את לוח האליפויות ללקוח | SERVER_LEADERBOARD |
| 7 | שרת | שולח ללקוח תפריט:<br>1. רענן<br>2. לחזור לתפריט הראשי | SERVER_LEADERBORAD_MENU |
| 8 | לקוח 1 | בוחר לרענן את לוח האליפויות | CLIENT_REFRESH |
| 9 | שרת | מעדכן את לוח האליפויות מהקובץ ושולח אותו שוב | SERVER_LEADERBOARD |
| 10 | שרת | שולח ללקוח תפריט:<br>1. רענן | SERVER_LEADERBORAD_MENU |

| | | | |
|---|---|---|---|
| | | 2. לחזור לתפריט הראשי | |
| **11** | לקוח 1 | בוחר לחזור לתפריט הראשי | CLIENT_MAIN_MENU |
| **12** | שרת | מציג לו את האפשרויות הבאות:<br>1. לשחק נגד שחקן אחר (אם מחובר)<br>2. לשחק מול "המחשב"<br>3. לצפות בלוח האליפויות<br>4. להתנתק מהשרת | SERVER _MAIN_ MENU |
| **13** | לקוח 1 | בוחר להתנתק מהשרת | CLIENT_DISCONNECT |
| **14** | שרת | השרת ממשיך לרוץ עד שמי שהריץ אותו מכבה אותו | |

## הודעות התקשורת

בתרגיל הזה, אתם תצטרכו לממש פרוטוקול מעל TCP.

מבנה ההודעות יהיה מבוסס טקסט. הודעה היא מערך תווים. המערך אינו מחרוזת, משום שהוא אינו מסתיים בתו '\0', ורשאי להכיל '\0'.

ההודעה מורכבת משני שדות, שמופרדים באמצעות התו נקודותיים (':').

<message_type>:<param_list>\n

1. message_type – סוג ההודעה. השדה הזה משמש את התוכנה כדי להבחין בין הודעות. כך ניתן להפעיל לוגיקה מתאימה לכל סוג הודעה.
2. param_list – רשימת פרמטרים מופרדים על ידי התו נקודה-פסיק (';').
   <param1>;<param2>;<param3>
   מספר הפרמטרים אינו קבוע.
3. '\n' – התו שמציין את סיום ההודעה. השדה הזה משמש את התוכנה כדי לזהות את סוף ההודעה.

אם יש 0 פרמטרים, ישלח השדה <message_type> בלבד, ללא נקודותיים (':').

הפרמטרים נשלחים בפורמט human readable. כלומר, גם כאשר הפרמטר מציין מספר, ישלח התו שמציין את המספר הזה, ולא תו שערך ה-ascii שלו שווה למספר. לדוגמא, אם הפרמטר הראשון הוא 1, ישלח התו '1' ולא התו '\1'.

להלן ההודעות אותן תידרשו להגדיר. אין להגדיר הודעות נוספות.

| שולח | message_type | תיאור | פרמטרים |
|---|---|---|---|
| **לקוח** | CLIENT_ REQUEST | הלקוח שולח לשרת את השם המשתמש | שם המשתמש |
| | CLIENT_MAIN_MENU | לקוח מבקש לעבור לתפריט הראשי | - |
| | CLIENT_CPU | לקוח רוצה לשחק נגד השרת | - |
| | CLIENT_VERSUS | לקוח רוצה לשחק נגד לקוח אחר | - |
| | CLIENT_LEADERBOARD | לקוח רוצה לצפות בלוח האליפויות | - |
| | CLIENT_PLAYER_MOVE | הלקוח בחר מהלך מבין: אבן, נייר, מספריים, לטאה או ספוק | שם המהלך שנבחר כמחרוזת: ROCK, PAPER, SCISSORS, LIZARD, SPOCK |

| | | | |
|---|---|---|---|
| | CLIENT_REPLAY | לקוח רוצה לשחק שוב (מול השרת או מול אותו לקוח). נשלח בסיום משחק | - |
| | CLIENT_REFRESH | הלקוח מבקש לבדוק אם יש עדכון בלוח האליפויות בזמן שצופים בו. אם יש שינוי הלקוח מעוניין לקבל את הלוח העדכני ואם לא הוא מעוניין בכך שיאמר לו שאין שינוי | - |
| | CLIENT_DISCONNECT | הלקוח מעוניין להתנתק מהשרת | - |
| **שרת** | SERVER_MAIN_MENU | השרת רוצה שהלקוח יציג למשתמש את התפריט הראשי (ראה בהמשך) | - |
| | SERVER_ APPROVED | השרת אישר את התחברותו של הלקוח | - |
| | SERVER_DENIED | השרת דחה את בקשת ההתחברות של הלקוח מסיבה כלשהי (אולי מטפל כבר בשני לקוחות) | הסיבה שבגללה הבקשה נדחתה כמחרוזת. |
| | SERVER_INVITE | השרת שולח ללקוח שעומד להתחיל משחק מול לקוח אחר ושולח בהודעה את שם המשתמש של היריב | שם הלקוח האחר |
| | SERVER_PLAYER_MOVE_REQUEST | השרת מבקש מהלקוח לבחור מהלך | - |
| | SERVER_GAME_RESULTS | תוצאות המשחק | שם הלקוח האחר (אם יש)<br>מה הוא שיחק<br>מה אתה שיחקת<br>מי ניצח |
| | SERVER_GAME_OVER_ MENU | השרת רוצה שהלקוח יציג למשתמש את תפריט סוף המשחק (ראה בהמשך) | - |
| | SERVER_OPPONENT_QUIT | השרת מודיע ללקוח שהלקוח האחר עמו שיחק לא מעוניין לשחק שוב | שם הלקוח האחר |
| | SERVER_NO_OPPONENTS | השרת מודיע ללקוח שאין לקוחות הפנויים למשחק כרגע | - |
| | SERVER_LEADERBOARD | השרת מעביר ללקוח את לוח האליפויות | הפרמטרים הם המידע של לוח האליפויות (הפורמט של לוח האליפויות יפורט בהמשך) |

דוגמא להודעה מסוג CLIENT_PLAYER_MOVE:

"CLIENT_PLAYER_MOVE:ROCK"

דוגמא להודעה מסוג S%ERVER_ APPROVED:

"SERVER_ APPROVED"

# תיאור מפורט

## לקוח

אלא אם נאמר אחרת זמן ההמתנה לתגובה מהשרת יהיה 15 שניות.

1. תוכנת הלקוח תתחבר לשרת בפרוטוקול TCP בכתובת שצוינה בארגומנטי הקלט.
2. לאחר חיבור מוצלח, תירשם השורה הבאה למסך:
   Connected to server on <ip>:<port>
3. במידה והחיבור נכשל, תירשם למסך ההודעה הבאה הבאה:
   Failed connecting to server on <ip>:<port>.
   Choose what to do next:
   1. Try to reconnect
   2. Exit
4. במקרה של התנתקות פתאומית מהשרת או TIMEOUT בהמתנה לתגובה מהשרת לאחר ההתחברות יש להתנתק מהשרת ולהדפיס למסך ההודעה הבאה:
   Connection to server on <ip>:<port> has been lost.
   Choose what to do next:
   1. Try to reconnect
   2. Exit

   במקרה שהמשתמש בוחר באופציה 1 הלקוח ינסה להתחבר מחדש לשרת. אם המשתמש בחר באופציה 2 הלקוח יסיים את ריצתו לאחר שיסגור וישחרר את כל המשאבים שהקצה במהלך הריצה.
5. לאחר החיבור לשרת הלקוח ישלח לשרת את שם המשתמש בהודעת CLIENT_REQUEST וימתין לקבלת הודעת SERVER_APPROVED. אם אין מענה יש להתנתק מהשרת ולהציג את אותה ההודעה מ-4. במידה ומתקבלת הודעת SERVER_DENIED יש להתנתק מהשרת ולהדפיס למסך את ההודעה הבאה:
   Server on <ip>:<port> denied the connection request.
   Choose what to do next:
   1. Try to reconnect
   2. Exit
6. לאחר חיבור מוצלח לשרת וקבלת אישור על שם המשתמש הלקוח יציג למשתמש את התפריט של SERVER_MAIN_MENU:
   Choose what to do next:
   1. Play against another client
   2. Play against the server
   3. View the leaderboard
   4. Quit
7. במידה והמשתמש בחר באופציה 1 השרת יחפש עוד לקוח שבחר באופציה זו ויתחיל בין הלקוחות משחק. במידה והוא מוצא לקוח שכזה השרת שולח לשניהם הודעת SERVER_INVITE ומתחיל ביניהם משחק. במידה ואין לקוח כזה השרת ישלח הודעת SERVER_NO_OPPONENTS. יש להמתין לתשובה במשך 30 שניות זאת משום שהשרת עצמו ימתין 15 שניות כדי לראות אם לקוח כלשהו מתחבר ורוצה לשחק. במקרה שאין שחקן אחר הלקוח יציג שוב את התפריט הראשי.

8. במידה והמשתמש בחר באופציה 2 יתחיל משחק מול השרת.
9. במידה והמשתמש בחר באופציה 3 הוא ימתין לקבלת הודעת SERVER_LEADERBOARD שמכילה את המידע על לוח האליפויות. לדוגמה:

| Name | Won | Lost | W/L Ratio |
|---|---|---|---|
| Noam | 13 | 2 | 7.5 |
| Shani | 6 | 9 | 0.667 |
| Dvir | 5 | 13 | 0.385 |

שימו לב שהלוח אליפויות ממוין לפי **היחס** בין ניצחונות להפסדים.
לאחר שהלקוח יציג למשתמש את לוח האליפויות הוא יציג לו את התפריט הבא:

Choose what to do next:
1. Refresh leaderboard
2. Return to the main menu

אם המשתמש בוחר באופציה 1 אז הלקוח ישלח הודעת CLIENT_REFRESH לשרת וימתין להודעת SERVER_LEADERBOARD חדשה.

10. במידה והמשתמש בחר באופציה 4 הלקוח ישלח לשרת הודעת CLIENT_DISCONNECT ויתנתק מהשרת. במקרה זה הלקוח יסיים את ריצתו לאחר שחרור כלל המשאבים שהקצה במהלך ריצתו.
11. במקרה של משחק נגד לקוח אחר או מול השרת כל פעם שהשרת יבקש מהמשתמש לבחור מהלך בהודעת SERVER_PLAYER_MOVE_REQUEST הלקוח יציג למשתמש את ההודעה הבאה:

Choose a move from the list: Rock, Paper, Scissors, Lizard or Spock:

וימתין עד שהמשתמש יבחר את אחת מהאופציות על ידי הקלדת השם של המהלך (הקלט יהיה case-**in**sensitive אבל מה שישלח לשרת תמיד יהיה באותיות גדולות כלומר המשתמש יכול להקליד Spock, SPOCK, spock, SpOcK וכו' אך מה שישלח לשרת תמיד יהיה רצף התווים SPOCK).
12. לאחר שליחת המהלך הלקוח ימתין לתוצאות המשחק מהשרת בהודעת SERVER_GAME_RESULTS והלקוח יציג למשתמש את התוצאות באופן הבא:

You played: <your_move>
<username> played: <opponent_move>
<winner> won!

כאשר your_move ו-opponent_move הם המהלכים של המשתמש והלקוח האחר בהתאמה, username הוא שם המשתמש של הלקוח האחר ו-winner הוא שם המנצח.
במידה והמשחק הוא מול השרת username יהיה המחרוזת "Server".
13. בתום המשחק השרת ישלח ללקוח תפריט סוף משחק בהודעת SERVER_GAME_OVER_MENU והלקוח יציג אותו למשתמש:

Choose what to do next:
1. Play again
2. Return to the main menu

באם המשתמש בחר באופציה 1 הלוגיקה של הזמנה למשחק חוזרת על עצמה רק אם גם הלקוח השני רוצה לשחק שוב, אחרת השרת ישלח הודעת SERVER_OPPONENT_QUIT והלקוח ידפיס למשך את ההודעה הבאה:

<username> has left the game!

כאשר username הוא שם המשתמש של היריב.

## השרת

אלא אם נאמר אחרת ההמתנה להודעה מהלקוח היא 15 שניות.

1. ה-thread הראשי של השרת יאזין לתקשורת נכנסת בפרוטוקול TCP על הפורט שצוין בארגומנט של שורת הפקודה.
2. ה-thread הראשי יבדוק באופן מחזורי (בין ההאזנות לפורט) אם נרשמה הפקודה exit ואם כן ינסה לסגור את כל ה-thread-ים שיצר, לשחרר את כל המשאבים שהקצה ולצאת.
3. כאשר לקוח מנסה להתחבר לשרת אם מספר הלקוחות שכבר מחוברים שווה למספר המקסימלי שהוגדר בהרצת השרת אז השרת ידחה את בקשת ההתחברות.
4. אם לא אז השרת יקבל את ההתחברות וייצור thread חדש עבור אותו לקוח שיטפל בו.
5. ה-thread החדש שנוצר ממתין להודעת CLIENT_REQUEST. השרת ישמור את שם המשתמש שהלקוח שולח להמשך וישלח ללקוח הודעת SERVER_APPROVED. מעתה אלא אם נאמר אחרת "השרת" מתייחס ל-thread שנוצר עבור אותו לקוח.
6. לאחר מכן השרת ישלח ללקוח הודעת SERVER_MAIN_MENU והלקוח יציג למשתמש את התפריט הראשי כפי שתואר קודם לכן. השרת ימתין (ללא הגבלת זמן) להחלטת הלקוח.
7. אם הלקוח בוחר להתנתק השרת יסיים (שוב, הכוונה ל-thread הספציפי שנוצר לאותו לקוח ולא לתוכנית השרת כולה).
8. אם הלקוח מעוניין לצפות בלוח האליפויות השרת יקרא מהקובץ Leaderboard.csv (אם הוא לא קיים אז יש ליצור קובץ חדש כזה לפי הפורמט שיוגדר בהמשך) בצורה מסונכרנת (למשל עם mutex) עם שאר ה-thread-ים של לקוחות אחרים וישלח את התוכן ללקוח בהודעת SERVER_LEADERBOARD.
9. השרת ימתין להחלטת הלקוח אם לרענן את הלוח או לחזור לתפריט הראשי. במידה ויחליט לרענן את הלוח השרת יבדוק יקרא מחדש מהקובץ וישלח את הלוח שוב.
10. במידה והלקוח מעוניין לשחק נגד השרת אז השרת מיד יגריל מהלך וישלח ללקוח הודעת SERVER_PLAYER_MOVE_REQUEST וימתין (ללא הגבלת זמן) להודעת CLIENT_PLAYER_MOVE. לאחר קבלת ההודעה מהלקוח השרת יחשב מי ניצח ומי הפסיד וישלח הודעת SERVER_GAME_RESULTS כאשר שם המשתמש שישלח יהיה "Server" וכן ישלח מה היה המהלך של השרת ומה המהלך של המשתמש והלקוח יציג זאת כמתואר קודם לכן. לאחר מכן ישלח הודעת SERVER_GAME_OVER_MENU וימתין להחלטת הלקוח אם לשחק שוב או לחזור לתפריט הראשי. אם יבחר לשחק שוב אז הלוגיקה חוזרת על עצמה ואם לא אז תשלח הודעת SERVER_MAIN_MENU.
11. במידה והלקוח מעוניין לשחק מול לקוח אחר השרת צריך לבדוק אם יש עוד לקוח שרוצה לשחק מול לקוח אחר. עליכם לממש מנגנון תקשורת בין ה-thread-ים של שני הלקוחות (יפורט בהמשך). לאחר שנמצא עוד שחקן השרת יבקש מכל אחד מהלקוחות את המהלך שלו בהודעת SERVER_PLAYER_MOVE_REQUEST וימתין 30 שניות לתשובות בהודעות CLIENT_PLAYER_MOVE משני הלקוחות. יחשב את תוצאת המשחק וישלח אותה בהודעת SERVER_GAME_RESULTS לשני הלקוחות. לאחר מכן ישלח לשניהם הודעת SERVER_GAME_OVER_MENU וימתין להחלטת כל אחד מהלקוחות אם הם מעוניינים לשחק שוב או לא. רק אם שני הלקוחות מעוניינים לשחק שוב יהיה עוד משחק, אם שני הלקוחות לא רוצים לשחק אז שניהם יחזרו לתפריט הראשי ואם רק אחד מהם רוצה אז הוא יקבל קודם הודעת SERVER_OPPONENT_QUIT ולאחר מכן את הודעת ה-SERVER_MAIN_MENU.

## לוח האליפויות

לוח האליפויות יישמר בקובץ בשם Leaderboard.csv (Comma Separated Values) כלומר הערכים בכל שורה יופרדו על ידי פסיק:

```
Name,Won,Lost,W/L Ratio
<username_1>,<W_1>,<L_1>,<W_1>/<L_1>
<username_2>,<W_2>,<L_2>,<W_2>/<L_2>
.
.
.
<username_N>,<W_N>,<L_N>,<W_N>/<L_N>
```

כאשר אתם נדרשים להציג את התוכן של הקובץ הוא ישלח ללקוח באותו פורמט בו הוא כתוב (CSV) אך יודפס למסך באופן הבא:

| Name | Won | Lost | W/L Ratio |
|---|---|---|---|
| <username_1> | <W_1> | <L_1> | <W_1>/<L_1> |
| <username_2> | <W_2> | <L_2> | <W_2>/<L_2> |
| . | | | |
| . | | | |
| . | | | |
| <username_N> | <W_N> | <L_N> | <W_N>/<L_N> |

**שימו לב**: יש שני tab-ים בין כל ערך.

את היחס בין ניצחנות להפסדים יש לשמור עם דיוק של 3 ספרות אחרי הנקודה עם עיגול מתאים (0,1,2,3,4 מתעגלים למטה ו-5,6,7,8,9 מתעגלים למעלה).

את הגישה לקובץ יש לסנכרן בעזרת semaphore ו-mutex עם שמות כך שניתן יהיה לגשת אליהם מכל thread בתהליך. ה-semaphore ישמש אותנו כדי לאפשר לפתוח את הקובץ לקריאה בלבד באופן מקבילי וה-mutex ישמש אותנו כדי לנעול אותו לכתיבה.

כאשר נרצה לפתוח את הקובץ לקריאה בלבד, נתפוס את ה-mutex ואז את ה-semaphore ולאחר מכן נשחרר מיד את ה-mutex. בצורה הזאת thread-ים אחרים גם יוכלו לפתוח את הקובץ לקריאה בלבד.

כאשר נרצה לפתוח את הקובץ לכתיבה כדי לעדכן אותו יהיה עלינו לתפוס את ה-mutex ולאחר מכן לתפוס את ה-semaphore ולא נשחרר את ה-mutex אלא נמתין עד שהמונה של ה-semaphore יגיע ל-1 (כלומר רק אנחנו מחזיקים אותו). בצורה זאת כל מי שרוצה לפתוח את הקובץ לקריאה או לכתיבה יצטרך לחכות שנסיים לכתוב.

<u>יש לעדכן את לוח האליפויות בסוף כל משחק כאשר כל thread יעדכן את הנתונים של הלקוח שלו.</u>

## תקשורת בין thread-ים בשרת

כאשר שני לקוחות רוצים לשחק זה עם זה ה-thread-ים שמטפלים בהם בשרת צריכים לתקשר. הדבר יעשה באופן הבא:

1. השרת יבדוק אם קיים קובץ GameSession.txt ואם לא אז ייצור אחד כזה (יש לסנכרן את היצירה והבדיקה עם mutex כדי שלא יהיה מצב שהקובץ ייווצר פעמיים).
2. אם הוא קיים אז נדע שאפשר לשחק מול ה-thread שפתח אותו ונבקש מהלקוח את המהלך שלו.
3. נרשום את המהלך לקובץ (יש לסנכרן את הפעולה עם mutex).
4. ה-thread השני יקרא את המהלך שלנו ויכתוב את שלו.
5. ה-thread שלנו יקרא את המהלך של היריב.
6. שני ה-thread-ים יחשבו את תוצאת המשחק וישלחו אותה ללקוחות.
7. שני ה-thread-ים יעדכנו את לוח האליפויות בהתאם לתוצאה.
8. ה-thread שיצר את GameSession.txt ימחק אותו.

אם הלקוחות בוחרים לשחק שוב אז התהליך חוזר על עצמו.

## סיום התכונה כאשר אין שגיאות

התוכנה תחזיר 0 אם הסתיימה ללא שגיאות.

כאשר אין שגיאות, התוכנה צריכה להסתיים באופן מסודר:

- אין לצאת מהתוכנה כאשר יש חוטים משניים פתוחים. יש לסמן להם סגירה ורק אם הם לא מסיים בעצמם לאחר 15 שניות מותר להשתמש ב-TerminateThread על מנת לסגור את החוטים.
- אין לצאת מהתוכנה כאשר יש handles פתוחים.
- אין לצאת מהתוכנה כאשר יש זיכרון דינאמי שלא שוחרר.
- אין לצאת מהתוכנה כאשר יש קבצים פתוחים.
- אין לצאת מהתוכנה כאשר יש sockets פתוחים.
- אין לצאת מהתוכנה לפני שקוראים ל-WSACleanup.

## טיפול בשגיאות

יש לבדוק את הצלחה של כל פונקציה שעלולה להיכשל (הקצאת זיכרון, פתיחת קבצים, יצירת חוט, פעולות על sockets וכו').

במקרה של שגיאה כלשהי, כגון כישלון בהקצאה דינאמית או כישלון בפתיחת תהליך, התוכנה תדפיס למסך הודעת שגיאה בעלת משמעות. היציאה מהתוכנית תיעשה בצורה מסודרת במקרה של שגיאה ברמת ה-thread שבו השגיאה התרחשה – כלומר באותו thread שהשגיאה התרחשה יש לשחרר את כלל המשאבים שהוקצו במיטב יכולתכם (Best Effort) ולסיים את הריצה של שאר החוטים באמצעות TerminateThread. שימו לב שבמקרה של הצלחה **כל** החוטים צריכים לשחרר את המשאבים שלהם.

## כתובות

כדי להתחבר לתוכנת שרת שרצה באותו מחשב כמו תוכנת הלקוח, השתמשו בכתובת ה-IP ל-localhost: 127.0.0.1.

אם תרצו, תוכלו להתחבר למחשב אחר. תוכלו לגלות מה כתובת ה-ip של מחשב על ידי הרצת הפקודה ipconfig ב-cmd באותו המחשב. בצעו ping ממחשב אחד לאחר, כדי לגלות אם הם רואים אחד את השני.

עבור מספר port, השתמשו במספר כלשהו בן 4 ספרות, למשל 8888.

## ריבוי פרויקטים באותו solution

במודל יש מדריך, שמסביר איך לעבוד עם מספר פרויקטים באותו ה-solution. המדריך נמצא במדור How-to.

בהצלחה בתרגיל,

*"Live long and prosper"*