# תרגיל בית מעשי מס' 2

להגשה עד 13.4.2011

שימו לב: באתר הקורס תוכלו למצוא מסמך עם הנחיות מפורטות לגבי הגשת תרגילים מעשיים. על כל התרגילים המעשיים שתגישו הסמסטר לעמוד בהנחיות הללו.

## מטרה

בתרגיל זה נרחיב את תרגיל בית מס' 1. נתכנן פרוטוקול אפליקציה וניממש אפליקציית רשת במבנה שרת/לקוח. האפליקציה תממש גרסה פשוטה של המשחק Nim:

http://en.wikipedia.org/wiki/Nim

פתרון התרגיל מורכב משתי תוכנות. הלקוח (client) – מאפשר משחק אינטראקטיבי לשחקן בודד. השרת (server) – מנהל את המשחק בין מספר לקוחות.

### המשחק

בסעיף זה נתאר את חוקי המשחק עבור גרסה פשוטה של Nim (זהה לגרסה של תרגיל 1, למעט האפשרות ליותר משני שחקנים).

המשחק מיועד לשני משתתפים ומעלה. למשחק יש פרמטר $N$ (תוכלו להניח $1 \leq N \leq 2000$), ופרמטר בוליאני *IsMisere*.

בתחילת המשחק, ישנן 3 ערימות *(heaps)* $A, B, C$ המכילות $n_A, n_B, n_C$ קוביות בהתאמה, $(1 \leq n_A, n_B, n_C \leq N)$.

בכל תור לוקח אחד המשתתפים מס' קוביות כרצונו (ולפחות אחת) מאחת הערימות.

במשחק רגיל *(IsMisere == false)* המנצח הוא האחרון שלוקח קוביות מהלוח, ושאר השחקנים מפסידים. במשחק *Misere* (כלומר *IsMisere == true*) המפסיד הוא האחרון שלוקח קוביות מהלוח, ושאר השחקנים מנצחים.

מספר השחקנים הוא $2 \leq p \leq 9$.

להלן דוגמה (לקוחה מ – *Wikipedia*) למשחק עם שני שחקנים שבו $n_A = 3, n_B = 4, n_C = 5$ ו - *IsMisere == false*:

(Example taken from http://en.wikipedia.org/wiki/Nim)

```
A B C

3 4 5   Bob takes 2 from A
1 4 5   Alice takes 3 from C
1 4 2   Bob takes 1 from B
1 3 2   Alice takes 1 from B
1 2 2   Bob takes entire A heap, leaving two 2s.
```

```
0 2 2   Alice takes 1 from B
0 1 2   Bob takes 1 from C leaving two 1s. (In misère play she would
        take 2 from C leaving (0, 1, 0).)
0 1 1    Alice takes 1 from B
0 0 1   Bob takes entire C heap and wins.
```

## הלקוח

תוכנית הלקוח מיועד לתקשורת בין השחקן לבין תוכנית השרת. הלוגיקה שבה מיועדת אך ורק לתקשורת עם המשתמש ועם תוכנית השרת. למשל, התוכנית לא "זוכרת" את גדלי הערימות של המשחק.

שורת הפקודה להרצה היא:

```
nim [hostname [port]]
```

כאשר *hostname* ו-*port* הם פרמטרים אופציונאליים. ערך ברירת המחדל הוא localhost = *hostname*, *port* = 6423. לא ניתן לספק *port* ללא *hostname*. הפרמטר *hostname* יכול להיות שם או כתובת IP.

למשל:

```
nim nova.cs.tau.ac.il 45678
```

הלקוח מתחבר לשרת בכתובת והפורט הנתונים.

לתוכנת הלקוח שני מצבים, שחקן וצופה (ראו פירוט בהמשך). הקצאת המצב נעשית ע"י השרת בזמן ההתחברות, ומשתנה מדי פעם לפי המוגדר בהמשך.

עם ההתחברות, הלקוח מקבל מהשרת את הערך של IsMisere ומציג אותו למשתמש באופן הבא:

```
This is a Misere game
```

או לחילופין:

```
This is a Regular game
```

לאחר מכן הוא מקבל את מספר השחקנים $p$ ומס' הלקוח שהוקצה ללקוח ומציג אותם:

```
Number of players is #
You are client #
```

כאשר # מוחלף בערכים המתאימים.

לסיום הוא מקבל מהשרת הודעה לגבי הסטטוס שלו במשחק (שחקן / צופה) ומדפיס בהתאם:

```
You are playing –OR– You are only viewing
```

במהלך המשחק הלקוח מגיב לאירועים הבאים:

1. קבלת עדכון מהשרת לגבי גדלי הערימות הנוכחיים. יודפס:

```
Heap sizes are #, #, #
```

(כאשר # מוחלף בערכים המתאימים.)

2. אם המשחק נגמר, הלקוח מקבל חיווי מהשרת לגבי זהות המנצח. אם הלקוח הוא אחד המשתתפים במשחק יודפס:

```
You lose! –OR- You win!
```

אם הלקוח הוא צופה יודפס:

```
Game over!
```

.

3. הלקוח יכול לקבל הודעה שזהו תורו, והוא מתבקש להזין את הצעד הבא שלו ל – console. מודפס:

```
Your turn:
```

4. קבלת קלט מהמשתמש ב – stdin. הקלט יכול להיות:
   a. Q עבור Quit, ליציאה מהמשחק
   b. מהלך במשחק - תו המייצג את הערימה הנבחרת (A, B, C) ואחריו מספר שלם המייצג את מספר הקוביות שברצון הלקוח להוריד מהערימה שבחר.
   c. הודעה המיועדת לשחקנים האחרים. הפורמט שלה הוא:

```
MSG # message
```

כאשר # הוא מספר הלקוח אליו מיועדת ההודעה (0 עבור broadcast לכל השחקנים והצופים) ולאחר מכן ההודעה עצמה, הנגמרת בסוף השורה.

תגובת הלקוח:

   a. אם הקלט היה Q, תוכנת הלקוח מסיימת את ריצתה מייד
   b. אם הקלט הוא מהלך, וזהו תורו של השחקן, המהלך נשלח לשרת. אחרת מודפס:

```
Move rejected: this is not your turn
```

   c. אם הקלט היה הודעה, היא נשלחת אל השרת מייד.

שים לב שהלקוח מוכן לקבל קלט מ – stdin בכל רגע, ולא רק כשזהו תורו של השחקן.

5. קבלת הודעה ממשתמש אחר והדפסתה בפורמט:

```
#: message
```

כאשר # הוא מספר הלקוח השולח.

6. קבלת אישור מהשרת שהמהלך האחרון שהוזן נקלט, או לחילופין התראה שהמהלך היה לא חוקי: למשל הלקוח ניסה להוריד 5 קוביות מערימה שהיו בה רק 3 קוביות. במקרה של מהלך לא חוקי, הלקוח מפסיד את התור, ולא מקבל הזדמנות להכניס קלט נוסף. יודפס:

```
Illegal move –OR- Move accepted
```

7. אם הלקוח הוא צופה הוא עשוי לקבל מהשרת הודעה על שינוי הסטטוס שלו לשחקן (ראה בהמשך). יודפס:

```
You are now playing!
```

8. בכל שלב על הלקוח לזהות גם ניתוק מהשרת. במקרה כזה, המשחק מסתיים ומודפס:

```
Disconnected from server
```

9. כשלקוח חדש מנסה להתחבר לשרת הוא יכול לקבל הודעה שנדחה מפני שיש כבר יותר מדי לקוחות מחוברים (ראה בהמשך). במקרה זה המשחק מסתיים ויודפס:

```
Client rejected: too many clients are already connected
```

ההדפסות למסך חייבות להיות בדיוק כפי שמתואר בסעיף זה.

## השרת

השרת מנהל משחק בודד. השרת "זוכר" את גדלי הערימות, מנהל את התורות, מקבל צעדי משחק מהלקוחות ומעביר ביניהם הודעות.

שורת הפקודה להרצה היא:

```
nim-server p N IsMisere [port]
```

כאשר $p, \mathrm{N}$ ו – IsMisere הם הפרמטרים למשחק המתוארים לעיל. IsMisere יקבל 1 או 0. הפורט להקשבה הוא פרמטר אופציונלי עם ערך ברירת מחדל 6423.

למשל:

```
nim-server 5 12 0 45678
```

לאחר שאותחל, השרת מגריל את גדלי הערימות התחיליים $n_A, n_B, n_C$, ומחכה להתחברות הלקוח הראשון.

אל השרת יכולים להתחבר בו זמנית עד 9 לקוחות. לקוחות נוספים שינסו להתחבר כשיש כבר 9 לקוחות מחוברים ידחו וידפיסו הודעה מתאימה כמתואר למעלה.

לכל היותר p מהלקוחות המחוברים בכל רגע הם שחקנים והשאר צופים. ההבדל בין שחקן לצופה הוא שתמיד תורו של שחקן כלשהו, ולעולם לא תור של צופה. השרת מחליט מיהו שחקן ומיהו צופה באופן הבא: כשמגיע לקוח חדש הוא מוגדר להיות צופה אם יש כבר p שחקנים, ומוגדר להיות שחקן אם יש פחות מ – p שחקנים.

כשלקוח חדש מתחבר מוקצה לו מספר הלקוח הנמוך ביותר שעדיין לא היה בשימוש. הלקוח הראשון שמתחבר הוא מס' 1. תוכלו להניח שבשום משחק לא יתחברו יותר מ – 25 לקוחות שונים (כלומר, מספרי הלקוחות הם תמיד בטווח $1, \ldots, 25$ ).

1. עם ההתחברות נשלחים ללקוח הפרטים המפורטים למעלה, גדלי הערימות, והודעה על כך שתורו, במידת הצורך.

2. כשלקוח מתנתק:
   a. מספר הלקוח שלו מתפנה.
   b. המשחק יושהה אם אין יותר לקוחות מחוברים עד להתחברות לקוח חדש.
   c. אם ישנם צופים והלקוח שהתנתק הוא שחקן, הצופה בעל מספר הלקוח הנמוך ביותר יהפוך להיות שחקן פעיל ויקבל הודעה מתאימה.
3. כשלקוח שולח הודעה על מהלך חוקי, בתורו, השרת מוריד בהתאם קוביות מן הערימה שנבחרה, ושולח לשחקן אישור על כך שהמהלך התקבל. לכל הלקוחות תשלח הודעה על ערכי הערימות המעודכנים. השרת בוחר את השחקן הבא שתורו לשחק (מתואר למטה), ומודיע לו על כך.
4. אם לקוח שולח הודעה על מהלך לא חוקי, או מנסה לשלוח מהלך שלא בתורו, הוא מקבל הודעה על צעד לא חוקי. לקוח שזה היה תורו ושולח מהלך לא חוקי מפסיד את התור ולא מקבל הזדמנות להכניס קלט חדש.
5. ניהול התורות: השרת מנהל סבב ציקלי של התורות בין כל השחקנים הפעילים (לא כולל הצופים). כלומר, נניח שיש 3 שחקנים מחוברים, סדר התורות יהיה ... ,1, 2, 3, 1, 2, 3.
   a. אם יש רק שחקן אחד מחובר, הוא יבצע תורות בזה אחר זה.
   b. אם אין לקוחות מחוברים המשחק מושהה עד להתחברות לקוח חדש, שיהפוך לשחקן ויקבל הודעה שזהו תורו מייד כשיתחבר.
   c. שימו לב ששחקן יכול לעזוב את המשחק כשזהו תורו, לפני ששלח מהלך. במקרה זה התור עובר לשחקן הבא.
   d. כששחקן חדש מצטרף (לקוח חדש מצטרף במעמד "שחקן", או שחקן עוזב ואז אחד הצופים מקבל הודעה שהפך לשחקן) הוא נכנס לסוף התור – כלומר, הוא יקבל את התור אחרי שכל השחקנים שנמצאים בתור כרגע יקבלו הזדמנות לשחק. אם אין שחקנים פעילים – השחקן החדש מקבל את התור מייד.
6. כשלקוח שולח הודעה ללקוח אחר או לכולם, ההודעה מופצת בהתאם. אם ההודעה נשלחת ללקוח שאינו מחובר, השרת מתעלם ממנה.
7. תוכנית השרת מסיימת את ריצתה כאשר אין יותר קוביות באף ערימה. היא תשלח את הודעת העדכון האחרונה לכל לקוח, שמודיעה שכל הערימות בגודל 0, ולאחר מכן תשלח לכל לקוח הודעה אם ניצח או הפסיד (בהתאם לפרמטר IsMisere).

לעולם לא נשלחת הודעה ללקוח שאינו מחובר.

## דוגמת ריצה

בצד השרת:

```
nim-server 2 6 0
```

לקוח ראשון (בפונט **ירוק** – קלט מהמשתמש):

```
nim
This is a Regular game
Number of players is 2
You are client 1
You are playing
Heap sizes are 3, 4, 5
Your turn:
```

```
MSG 2 Good luck!
A 3
Move accepted
Heap sizes are 0, 4, 5
Heap sizes are 0, 0, 5
Your turn:
C 5
Move accepted
Heap sizes are 0, 0, 0
You win!
```

לקוח שני (בהנחה שהתחבר מייד אחרי לקוח 1, לפני שלקוח 1 שלח את המהלך הראשון שלו):

```
nim
This is a Regular game
Number of players is 2
You are client 2
You are playing
Heap sizes are 3, 4, 5
1: Good luck!
Heap sizes are 0, 4, 5
Your turn:
B 4
Move accepted
Heap sizes are 0, 0, 5
Heap sizes are 0, 0, 0
You lose!
```

לקוח שלישי (צופה, בהנחה שהתחבר מייד אחרי לקוח 2, לפני שלקוח 1 שלח את המהלך הראשון שלו):

```
nim
This is a Regular game
Number of players is 2
You are client 3
You are only viewing
Heap sizes are 3, 4, 5
Heap sizes are 0, 4, 5
Heap sizes are 0, 0, 5
Heap sizes are 0, 0, 0
Game over!
```

## דרישות התרגיל

ראשית, עליכם לתכנן פרוטוקול אפליקציה מתאים שיעבוד מעל TCP. לאחר מכן, ממשו אותו כפי שנלמד בתרגול. עליכם לרבב בין לקוחות באמצעות שימוש בפקודה select, כפי שלמדנו בתרגיל (זכרו שגם את stdin ו – stdout אפשר להכניס ל – select).

את פרוטוקול האפליקציה יש לתעד בבירור, באופן שיאפשר לכל אדם לממש לקוח או שרת ש"ידברו" עם התוכנות שהגשתם.

הדגש בבדיקת הקוד שתגישו ינתן כמובן למימוש התקשורת ברשת. על המימוש להיות יעיל ורובוסטי (robust). אל תשכחו לבדוק שגיאות בערכי חזרה מפונקציות ולטפל בהם בהתאם.

למקרה שתבחרו להגיש בסביבת nova: ייתכן שזוגות רבים יבדקו את הפתרון שלהם על nova בו-זמנית. לכן, מומלץ להשתמש בפורט שרת שאינו פורט ברירת המחדל בעת בדיקת הקוד.