

## תרגיל בית מעשי מס' 1

להגשה עד 15.3.2011

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 נתכנן פרוטוקול אפליקציה ונממש 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, ומשחק מול השחקן שמשתמש בתוכנת הלקוח.

בתרגיל מעשי מספר 2 נרחיב את האפליקציה כך שתתמוך במספר שחקנים.

### המשחק

בסעיף זה נתאר את חוקי המשחק עבור גרסה פשוטה של Nim.

המשחק מיועד לשני משתתפים. 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. במשחק *Misere* (כלומר  $IsMisere == true$ ) המפסיד הוא האחרון שלוקח קוביות מהלוח.

להלן דוגמה (לקוחה מ-Wikipedia) למשחק שבו  $n_A = 3, n_B = 4, n_C = 5$  -  $IsMisere == false$ :

(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
```

עמוד 1 מתוך 4

רשתות תקשורת מחשבים, סמסטר ב' 2010/11  
ביה"ס למדעי המחשב, אוניברסיטת ת"א

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 *hostname = localhost*, *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.

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
```

במהלך המשחק הלקוח מבצע את הצעדים הבאים אחד אחרי השני בלולאה, עד לסיום המשחק:

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

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

(כאשר # מוחלף בערכים המתאימים.)

2. אם המשחק נגמר, הלקוח מקבל חייווי מהשרת לגבי זהות המנצח. יודפס:

```
I win! -OR- You win!
```

3. אחרת, הלקוח מתבקש להזין את הצעד הבא שלו ל – console. מודפס:

```
Your turn:
```

הקלט הוא תו המייצג את הערימה הנבחרת (A, B, C) או Q עבור Quit ליציאה מהמשחק. אם הקלט אינו Q, התוכנית מקבלת גם מספר שלם המייצג את מספר הקוביות שברצון הלקוח להוריד מהערימה שבחר.

4. אם הקלט היה Q, תוכנת הלקוח מסיימת את ריצתה מייד, אחרת הצעד של השחקן נשלח לשרת.

רשתות תקשורת מחשבים, סמסטר ב' 2010/11  
ביה"ס למדעי המחשב, אוניברסיטת ת"א

5. הלקוח מקבל אישור מהשרת שהמהלך 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
```

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

```
Disconnected from server
```

ההדפסות למסך חייבות להיות בדיוק כפי שמתואר בסעיף זה.

### השרת

השרת מנהל משחק בודד. השרת "זוכר" את גדלי הערימות, מקבל צעדי משחק מהלקוח, ומשחק אחרי  
כל צעד של הלקוח.

שורת הפקודה להרצה היא:

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

כאשר  $N$  – IsMisere הם הפרמטרים למשחק המתוארים לעיל. IsMisere יקבל 1 או 0. הפורט  
להקשבה הוא פרמטר אופציונלי עם ערך ברירת מחדל 6423.

למשל:

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

לאחר שאותחל, השרת מגריל את גדלי הערימות התחיליים  $n_A, n_B, n_C$ , ומחכה להתחברות מהלקוח.  
לאחר התחברות המשחק מתחיל. השרת מגיב להודעות מהלקוח כפי שפורט בסעיף הקודם.

אחרי תורו של הלקוח, אם המשחק לא הסתיים מגיע תורו של השרת. תוכלו לממש לשרת לוגיקת משחק  
לפי בחירתכם, ואין חובה לממש אסטרטגיה "חכמה" (למרות שאפשר, קראו בויקיפדיה).

תוכנית השרת מסיימת את ריצתה לאחר שהודיעה ללקוח על סוף המשחק, או לאחר שהלקוח התנתק.

### דוגמת ריצה

בצד השרת:

```
nim-server 6 0
```

בצד הלקוח (בפונט ירוק – קלט מהמשתמש):

```
nim
This is a Regular game
Heap sizes are 3, 4, 5
Your turn:
A 3
```

עמוד 3 מתוך 4

רשתות תקשורת מחשבים, סמסטר ב' 2010/11  
ביה"ס למדעי המחשב, אוניברסיטת ת"א

```
Move accepted
Heap sizes are 0, 0, 5
Your turn:
A 5
Illegal move
Heap sizes are 0, 0, 0
I win!
```

בדוגמה הזו, התחלנו עם  $n_A = 3, n_B = 4, n_C = 5$ , במשחק רגיל (IsMisere == false). הלקוח הוריד את כל הקוביות מערימה A, ואז השרת הוריד את כל הקוביות מערימה B. הלקוח נתן קלט לא חוקי, ולכן איבד תור. השרת ניצל זאת כדי להוריד את כל הקוביות מערימה C, ובכך ניצח.

### דרישות התרגיל

ראשית, עליכם לתכנן פרוטוקול אפליקציה מתאים שיעבוד מעל TCP. לאחר מכן, ממשו אותו כפי שנלמד בתרגול. האפליקציה סינכרונית, כלומר אפשר להשתמש בפקודות חוסמות.

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

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.