Homework Dry 3

Due date: Friday, 29.5.2015, 12:30 noon

Teaching assistant in charge: Arthur Kiyanovski arthurk@cs.technion.ac.il

the Q&A for the exercise will take place at a public forum Piazza only. Important: Please note, the forum is a part of the exercise. Important clarifications/corrections will also be published in the FAQ on the site. A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers
- Be polite, remember that course staff does this as a service for the students
- You're not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour
- When posting questions regarding hw1, put them in the hw1 folder

Please submit your answers printed – not printed works will not be checked.

שאלה 1

מחסום בגודל n (n-barrier) הינו מנגנון סינכרון בין חוטים המאפשר לקבוצת תהליכים (כאשר מספר התהליכים המשתתפים n ידוע מראש) לתאם ביניהם את שלבי הביצוע באלגוריתם מקבילי. למשל, אם לאלגוריתם המקבילי ישנם שני שלבים מקביליים (שלב א' ושלב ב'), כלומר, בשלב א' החוטים יכולים לבצע חישוב במקביל ובשלב ב' החוטים יכולים לבצע חישוב במקביל, אבל לשם נכונות האלגוריתם אסור לאף חוט שסיים את שלב א' של האלגוריתם להתחיל את שלב ב' לפני שכל החוטים האחרים סיימו את שלב א', אז דרוש להשתמש במחסום כנ"ל עבור כל החוטים המשתתפים במעבר בין שלב א' לשלב ב' של האלגוריתם.

סטודנט מימש את הפונקציה barrier_wait באמצעות משתנה תנאי ו- mutex. להלן הקוד שכתב:

```
struct Barrier {
   pthread_cond_t cond;
   pthread_mutex_t mutex;
   int n, count;
};

void barrier_init(struct Barrier *b, int n) {
  pthread_cond_init(&b->cond, NULL);
  pthread_mutex_init(&b->mutex);
  b->n = n;
  b->count = 0;
}

void barrier_wait(struct Barrier *b) {
  pthread_mutex_lock(b->mutex);
  b->count++;
  if (b->count < b->n)
     pthread_cond_wait(&b->cond, &b->mutex);
  if (b->count == n) {
    b->count = 0;
  }
  pthread_cond_signal(&b->cond);
  pthread_mutex_unlock(&b->mutex);
}
```

כאשר הסטודנט הריץ את האלגוריתם המקבילי שלו המשתמש במחסום שמימש **מספר פעמים**, נוכח לראות שהמחסום שלו לא ממומש נכון.

a. הדגם והסבר בעיה אפשרית שיכולות לנבוע ממימוש המחסום לעיל.
b. הצע תיקון למימוש המחסום, כך שהבעיה שתיארת בסעיף א' תיפתר.
c. ממש מחסום באמצעות mutex ו-(n-1) סמפורים בלבד (הראה רק את המימוש של barrier_wait, הנח ששאר הפונקציות והרשומה הנתונים מומשו עבורך).
d. בסעיף הקודם מימשת מחסום עם (n-1) סמפורים. מדוע לא ניתן היה לממש אותו עם סמפור יחיד? רמז: נסה להחליף את השימוש בסמפורים מהפתרון של סעיף קודם בסמפור יחיד ואז נסה למצוא בעיה עם המימוש.

שאלה 2

למתרגל במערכות הפעלה יש שעת קבלה פעם בשבוע. אם אף אחד לא בא לשאול אותו שאלות בזמן שעת הקבלה הוא גולש באינטרנט. אם סטודנטים באים לשאול שאלות, הם צריכים להסתנכרן בינם כך שיתקיימו התנאים הבאים:

1. רק סטודנט אחד שואל שאלה בכל רגע נתון.
2. סטודנט יכול לשאול שאלה רק אם המתרגל סיים לענות על השאלה הקודמת.

כמו כן, המשרד של המתרגל הוא קטן ולא יכול להכיל יותר מ –4 סטודנטים. סטודנט שמגיע ומגלה שהוא לא יכול להיכנס למשרד עוזב.

הפסאודו-קוד הבא מממש את הבעיה המוצגת. המתרגל מבצע את פרוצדורת TA(). סטודנט שמגיע לשעות הקבלה מבצע את פרוצדורת Student().

הניחו שמשתני תנאי ו-mutex-ים מאותחלים.

Global variables and Data

```
MAX_STUDENT=4;

students = 0;

questions = 0;

Lock lock;

Condition TA_cond;

Condition student_cond;

answering = FALSE;
```

```
procedure TA()
{
  while (true)
  {
     acquire(lock);
    while ((students == 0) || (questions == 0))
                        cond_wait(TA_cond, lock);
    questions --;
    answering = TRUE;
    release(lock);


    AnswerQuestion();


    acquire(lock);
    answering = FALSE;
    cond_signal(student_cond);
    release(lock);
   }
}


procedure Student()
{
```

```
20.     acquire(lock);
21.     if (students == MAX_STUDENT)
22.     {
23.       release(lock);
24.       return;
25.     }

26.     students++;

27.     release(lock);

28.     while (have questions to ask)
29.     {
30.       acquire(lock);
31.       while (answering)
32.         cond_wait(student_cond, lock);
33.       question ++;
34.       AskQuestion();
35.       cond_signal(TA_cond);
36.       release(lock);
37.     }

38.     acquire(lock);
39.     students--;
40.     release(lock);
41.   }
```

א. האם הפתרון הנ"ל מבטיח שסטודנט לא ישאל שאלה כאשר מתרגל עונה על שאלה של סטודנט אחר (מניעה הדדית בין מתרגל לסטודנט) ? אם כן נמק/י בפרוט, אם לא תן/י דוגמה.

ב. האם הפתרון הנ"ל מבטיח ששני סטודנטים לא ישאלו שאלה באותו זמן (מניעה הדדית בין הסטודנטים, תנאי 1 בשאלה)? אם כן נמק/י בפרוט, אם לא תן/י דוגמה.

ג. האם הפתרון הנ"ל מבטיח שסטודנט יכול לשאול שאלה רק אם המתרגל סיים לענות על השאלה הקודמת (תנאי 2 בשאלה)? אם כן נמק/י בפרוט, אם לא תן/י דוגמה. (שימו לב, ההבדל בין א' לג' הוא שכאן אנחנו שואלים האם מתרגל הספיק לענות על שאלה. יתכן שהוא פשוט עדיין לא התחיל לענות וסטודנט אחר כבר שואל שאלה – שזה נוגד תנאי 2).

ד. מה יקרה אם נחליף בין שורה 33 בקוד (question ++) לשורה 35 (cond_signal)?

ה. מה יקרה אם נחליף את cond_signal בשורה 15 ל cond_broadcast? נמקו.

ו. מה יקרה אם נוריד את שורות 38 ו 40 ? הסבירו.

שאלה 3

1. הסבר את ההבדל בין מנעול הוגן למנעול לא הוגן.
2. הסבר מהי הרעבה בהקשר של מנעולי סנכרון.
3. מה הקשר בין הרעבה במנעולי סנכרון לבין הוגנות המנעולים? הבא דוגמא להרעבה בהקשר זה.
4. הסבר מהו קיפאון (deadlock)?
5. האם קיפאון גורר הרעבה? אם כן הסבר, אם לא תן דוגמא.

6. מדוע הפונקציה pthread_cond_wait() מקבלת mutex כפרמטר?
7. מדוע משתנה תנאי מקושר לmutex? התייחס לשימוש במשתנה תנאי שראינו בתרגולים (מנעול קוראים כותבים) והסבר בקצרה מה היה משתבש אילו לא היה mutex מקושר למשתנה התנאי.