אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד, תשובות מחוץ לגיליון לא יבדקו.

שימו לב:

על תשובות ריקות יינתן 20% מהניקוד.

בהצלחה!

תאריך הבחינה: 6.3.2017
שם המורה: ד"ר מני אדלר
ד"ר אחיה אליסף
מר בני לוטטי
פרופ' אנדרי שרף
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשע"ז
סמסטר: א'
מועד: ב'
משך הבחינה: שלש שעות
חומר עזר: אסור

שאלה 1 (30 נקודות)

סעיף א.

תנאי הקדם - 2 נקודות. תנאי הבתר - 3 נקודות.

הערה על הפתרון: בתנאי הקדם אין לבדוק שהמשחק לא השתנה. המתודה מוגדרת היטב ויש לה ערך החזרה למקרה זה.

פתרון: בשביל לקצר, נניח כי קיימת מתודה isEmpty(int x, int y) המחזירה true אם"ם יש שם שחקן או גביע.

```
@PRE: a != null && dir != null  && checkBoundsValidity(a) && getPlayer(a) != null.
@POST:
    // Return value:
    @return > 0 && @return < size()
    dir == UP ? (for each i, s.t. : a.i - @return < i < a.i : isEmpty(i, a.j) == true)
    : dir == DOWN ? (for each i, s.t. : a.i < i < a.i + @return : isEmpty(i, a.j) == true)
    : dir == LEFT ? (for each j, s.t. : a.j - @return < j < a.j : isEmpty(a.i,j) == true)
    : (for each j, s.t. : a.j < j < a.j + @return : isEmpty(a.i, j) == true ) // RIGHT

    // Object's state did not change:
    && for each Location x, s.t. 0 <= x.i, x.j < size() : getPlayer(x) == @PRE(getPlayer(x))
    &&  gameEnded() == @PRE(gameEnded())
```

סעיף ב.
תשובה מלאה מורכבת משני חלקים:
1. הנחה על (או מצב נתון של) הלוח. בדוגמה שבפתרון: מיקומי שחקנים.
2. הסבר למה יכול להיות חבק במצב כזה אפשר במילים או בטבלת זמנים.

הסתפקות רק בשלב הראשון לא מסבירה למה יכול להיות חבק ולכן גררה הורדת ניקוד חלקי.
ירדו כל הנקודות על הסבר שגוי שלא מוביל לחבק. ציור מדויק גם התקבל.

פתרון: נניח ששחקן א' נמצא ב-(0,0) ושחקן ב' נמצא ב-(3,0), ובניהם אין שחקן אחר או גביע.
שימו לב שצריך מינימום שני תאים בין השחקנים.

זמן	שחקן א'	שחקן ב'
0	קורא למתודה עם הכיוון ימינה	קורא למתודה עם הכיוון שמאלה
1	נועל את (1,0)	נועל את (2,0)

סעיף ג.
טעויות נפוצות:
- סנכרון כל הלולאה כך שהמתודה בעצם סדרתית - ירדו כל הנקודות.
- אי שימוש ב resource ordering - ירדו 5 נקודות.
- הגדרת סדר על השחרור ולא על הנעילה - ירדו 4 נקודות.
- סידור עם hashcode שיכול להיות אקראי במקום במיקום. אמנם זה פותר חבק אבל יוצר בעיות אחרות - ירדו 3 נקודות.
- חבק - ירדו 2 נקודות.
- מקרי קצה חסרים - ירדה נקודה.
- שימוש ב tryAcuire. אם לא מצליח - לצאת. באופן זה אולי אין deadlock (תלוי מימוש), אבל המתודה לא עושה את מה שהיא אמורה לעשות - ירדו 4 נקודות.
- שימוש ב tryAcuire בלולאה עד שמצליח - זה busy wait - ירדו 2 נקודות.

פתרון: נגדיר סדר על הלוח לפי הסדר הטבעי של מטריצות. נשים לב שכאשר אנו מסתכלים ימינה או למטה - אנו מסתכלים לפי הסדר הטבעי. לכן, אם אנו רוצים לנעול מספר מינימלי של תאים (לא חובה כיוון שלא היתה דרישה כזו) אפשר לנעול מ-a תא-תא עד שנגיע לקצה הלוח או לתא המכיל שחקן אחר או גביע. עם זאת, כאשר אנו מסתכלים למעלה או שמאלה, אנו מסתכלים נגד הסדר הטבעי, ולכן נצטרך לנעול מהגבול העליון או השמאלי (בהתאמה), ועד המיקום של a.

פתרון פשוט וקצר יותר, אבל שתופס יותר מנעולים, הינו לתפוס את כל השורה במקרה של RIGHT ו-LEFT, ואת כל העמודה במקרה של UP ו-DOWN.

הערה: סדר שחרור מנעולים לא יכול לגרום לחבק, אלא אם כן בין שחרור מנעול א' לשחרור מנעול ב' - מתבצעת נעילה נוספת (מה שיכול להיות בקוד מורכב). להרחבה - ראו את התשובה השניה ב - http://stackoverflow.com/questions/1951275/would-you-explain-lock-ordering.
מסיבה זו ואחרות: מומלץ שסדר שחרור המנעולים יהיה הפוך מסדר תפיסתם, זוהי מתודולוגיה טובה ונכונה.

```
public int lookAhead(Location a, Direction dir) {
    if (!checkBoundsValidity(a)) throw new IllegalArgumentException();
    if (gameEnded()) return -1;
    Deque<Semaphore> locks = new LinkedList<>();
    Location current;
    try {
```

```
        switch (dir) {
            case RIGHT:
                lock(a, locks);
                if (getPlayer(a) == null) throw new IllegalArgumentException();
                current = a.move(Direction.RIGHT);
                while (checkBoundsValidity(current)) {
                    lock(current, locks);
                    if (current.equals(goblet) || getPlayer(current) != null) break;
                    current = current.move(Direction.RIGHT);
                }
                return current.j - a.j;
            case LEFT:
                for (int j = 0; j <= a.j; j++)
                    lock(new Location(a.i, j), locks);
                if (getPlayer(a) == null) throw new IllegalArgumentException();
                for (int j = a.j - 1; j >= 0; j--) {
                    current = new Location(a.i, j);
                    if (current.equals(goblet) || getPlayer(current) != null) return a.j - j;
                }
                return a.j + 1;
            case DOWN:
                lock(a, locks);
                if (getPlayer(a) == null) throw new IllegalArgumentException();
                current = a.move(Direction.DOWN);
                while (checkBoundsValidity(current)) {
                    lock(current, locks);
                    if (current.equals(goblet) || getPlayer(current) != null) break;
                    current = current.move(Direction.DOWN);
                }
                return current.i - a.i;
            case UP:
                for (int i = 0; i <= a.i; i++)
                    lock(new Location(i, a.j), locks);
                if (getPlayer(a) == null) throw new IllegalArgumentException();
                for (int i = a.i - 1; i >= 0; i--) {
                    current = new Location(i, a.j);
                    if (current.equals(goblet) || getPlayer(current) != null) return a.i - i;
                }
                return a.i + 1;
            default: throw new IllegalArgumentException();
        }
    } catch (InterruptedException e) {
        return -1;
    } finally {
        while (!locks.isEmpty()) locks.pop().release();
    }
}
```

```
private void lock(Location a, Deque<Semaphore> locks) throws InterruptedException {
   Semaphore semaphore = getLock(a);
   semaphore.acquire();
   locks.push(semaphore);
}
```

סעיף ד.

טעויות נפוצות:

- כתיבת קוד שכל מה שהוא עושה זה להעלות ולהוריד גרסה, מבלי לבדוק אם ה Segment שהתקבל חוצה את ה Segment-ים האחרים שברגע זה נבדקים - ירדו כל הנקודות (הקוד הזה פשוט לא עושה כלום).
- במקרה שהגרסה השתנתה - הבדיקה של ה Segment-ים לא התחילה מההתחלה - ירדה נקודה.
- במקרה שאחרי הלולאה של בדיקת ה-Sement-ים לא היתה בדיקה חוזרת של הגרסה - ירדו 2 נקודות.
- הגדרת גרסה ללא volatile או Atomic - ירדה נקודה.
- שימוש ב wait/notify ללא סנכרון - ירדה נקודה.
- חבק - ירדו 2 נקודות.
- סנכרון כל הלולאה למעשה הופכת את הקוד לסדרתי ולכן - ירדו 5 נקודות.
- אי סנכרון בין הוספה / הסרה לבין שינוי הגרסה - לא ירדו נקודות.

```
private final ConcurrentLinkedQueue<Segment> acquiredSegments;
private volatile int version = 0;
…
public void acquireSegment(Segment s) throws InterruptedException {
   LOOP:
   while (true) {
      int v = version;
      for (Segment other : segments) {
         if (other.intersect(s)) {
            synchronized (segments) {
               while (version == v) segments.wait();
               continue LOOP;
            }
         }
      }
      synchronized (segments) {
         if (v == version) {
            version++;
            segments.add(s); //operation in O(1)
            break;
         }
      }
   }
}
public void releaseSegment(Segment s) {
   segments.remove(s);
   synchronized (segments) {
      version++;
      segments.notifyAll();    } }
```

שאלה 2 (25 נקודות)

סעיף א (8 נקודות)

```
template<typename T> Terrain<T>::~Terrain()
{

	if (!array_)
		return;

	for (int i = 0; i < width_; i++)
	{
		delete array_[i];
	}

	delete array_;
}

template<typename T> Terrain<T>::Terrain(int n, int m)
{

	width_ = n;
	height_ = m;
	array_ = new T*[width_];
	for (int i = 0; i < width_; i++)
		array_[i] = new T[height_];
}

template<typename T> Terrain<T>::Terrain(const Terrain & other)
{
	width_ = other.width_;
	height_ = other.height_;
	array_ = new T*[width_];

	for (int i = 0; i < width_; i++)
	{
		array_[i] = new T[height_];
		for (int j = 0; j < height_; j++)
		{
			array_[i][j] = other.array_[i][j];
		}
	}
}
```

אי הקצאה של מערך דו ממדי , אי שימוש בלולאה, אי שימוש בT* עבור מערך של מערכים: הורדו בין 2 ל5 נק.

סעיף ב (9 נקודות)

```
void main()
{
	int n = 10, m = 10;
	Game<GameFigure*> game1;

	GameFigure **array = new GameFigure*[100];
	for (int i = 0; i < 100; i++)
		if (i%2)
			array[i] = new Cavalier();
		else
			array[i] = new Knight();

	game1.Init(n, m, array);
}
```

אי הקצאה נכונה במחסנית ובערמה כנדרש: עד 4 נק.
אי שימוש נכון בTEMPLATE: הורדו 2 נק.

סעיף ג (8 נקודות)

```
Terrain& operator=(Terrain &&other) { steal(other); return *this; }

template<typename T> void Terrain<T>::steal(Terrain<T> &other)
{
	array_ = other.array_;
	width_ = other.width_;
	height_ = other.height_;

	other.array_ = nullptr;
}
```

בלבול בין אופרטור= לבין בנאי מעתיק: 2 נק.
אי שימוש נכון בSTEAL כדוג. העתקה והקצאה: 4 נק.

שאלה 3 (30 נקודות)

הערות כלליות:

- השאלה עסקה ביישום פשוט של תבנית שנלמדה באופן מפורט בכיתה.
- השאלה באה לבחון הבנה של התבנית הנתונה, ע"י כתיבת קוד המחולל אותה.
- מצופה מסטודנטים להיות מסוגלים לכתוב קוד בן מספר שורות.
- מצופה מסטודנטים בסוף הקורס להיות מסוגלים לכתוב קוד המפעיל ת'רד.
- מצופה מסטודנטים בסוף הקורס להבין שבמערכת מקבילית עם זיכרון משותף נדרש סנכרון.
- אי כתיבה של קוד כזה מעלה חשד כבד לשותפות קש בתרגילים.
- כל ניקוד מעל 0 על תשובה שהסתכמה במילים ניתן לפנים משורת הדי/

סעיף א (7 נקודות)

```
class BankState {
   Map<String,Integer> mapUser2Amount;


   BankState() {
      mapUser2Amount = new HashMap<String,Integer>();
   }


   public synchronized void addUser(String user) {
      if (!mapUser2Amount.containsKey(user))
         mapUser2Amount.put(user,0);
   }

   public synchronized int getUserAmount(String user) throws Exception {
      Integer amount = mapUser2Amount.get(user);
      if (amount == null)
         throw new Exception("Unknown user: " + user);
      else
         return amount;
   }


   public synchronized void updateAmount(String user, int amount) throws Exception {
      Integer currentAmount = mapUser2Amount.get(user);
      if (currentAmount == null)
         throw new Exception("Unknown user: " + user);
      else {
         int updatedAmount = currentAmount + amount;
         if (updatedAmount < 0)
            throw new Exception("Overdraft!");
         mapUser2Amount.put(user,updatedAmount);
   }
}
```

סעיף ב (9 נקודות)

```
public class LoginCommand implements Command<BankState> {
  //implementation of the login command

  String user;

  LoginCommand(String user) { this.user = user; }

  public Serializable execute(BankState bank) {
     bank.addUser(user);
     return “ack”;
  }
}

public class DepositCommand implements Command<BankState> {
  //implementation of the deposit command
  String user;
  int amount;

  DepositCommand(String user, int amount) { this.user = user; this.amount = amount; }

  public Serializable execute(BankState bank) {
     try {
        bank.updateAmount(user,amount);
        return “ack”;
     } catch (Exception e) {
        return “nak”;
     }
  }

}

public class TransferCommand implements Command<BankState> {
  //implementation of the transfer command
  String fromUser, toUuser;
  int amount;

  DepositCommand(String fromUser, String toUser, int amount) {
     this.fromUser = fromUser;
     this.toUser = toUser;
     this.amount = amount;
   }

  DepositCommand(String user, int amount) { this.user = user; this.amount = amount}
     try {
        bank.updateAmount(fromUser,-1 * amount);
        bank.updateAmount(toUser,amount);
        return “ack”;
     } catch (Exception e) {
        return “nak”;
     }
}
```

סעיף ג (6 נקודות)

```
public class BankServer {
  public static void main(String[] args) {

     BankState bank = new BankState(); //one shared object

      new Reactor (
            10, // num of threads
            7777, //port
            () -> new RemoteCommandInvocationProtocol<>(bank), //protocol factory
            () -> new ObjectEncoderDecoder //message encoder decoder factory
      ).serve();
   }
  }
}
```

סעיף ד (8 נקודות)

```
public class BankClients {
  public static void main(String[] args) {
    new Thread(getClient("Hilak", "Bilak")).start();
    new Thread(getClient("Bilak","Hilak")).start();
  }


  Runnable getClient(final String user1, final String user2)  {
     return new Runaable() {
       try (RCIClient c = new RCIClient(host, 7777)) {
         c.send(new LoginCommand(user1));
         c.send(new DepositCommand(user1,1000));
         while (true) {
            c.send(new TransferCommand(user1,user2,1000));
            String ans = (String)c.receive();
            if (ans.equals("nak")
               System.out.println("Transfer of 1000 NIS from " + user1 + " to " + user2 " was failed");
         }
      }
     };
  }
}
```

שאלה 4 (15 נקודות)

סעיף א (5 נקודות)

סעיף ב (5 נקודות)

סעיף ג (5 נקודות)