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)

בשאלה זו נידון במשחק לוח. הלוח בגודל N x N ובתחילת המשחק מפוזרים עליו אקראית גביע אחד ומספר שחקנים (ת'רדים). השחקנים יכולים בכל רגע נתון להפעיל את המתודה move - המזיזה שחקן צעד אחד לכיוון כלשהו (enum Direction), או lookAhead - המחזירה את המרחק (מספר הצעדים) לעצם הקרוב ביותר, או קצה הלוח בכיוון הנתון. עצם יכול להיות שחקן אחר או הגביע. למחלקה יש מתודות נוספות המוגדרות בממשק Board.

להלן מימוש חלקי של המערכת, אנא קראו בעיון את **כל הקוד** עד סופו בטרם תיגשו לענות:

```
public enum Direction { UP, DOWN, RIGHT, LEFT }
public enum Result { SUCCESS, FAIL, WIN, LOST }
public final class Player implements Runnable { … }

public final class Location implements Comparable<Location> {
   public final int i, j;
   public Location(int i, int j) { this.i = i; this.j = j; }
   @Override  public boolean equals(Object obj) { … }
   @Override public int compareTo(Location o) { … }
   public Location move(Direction direction) {
      switch (direction) {
         case RIGHT:
            return new Location(i, j + 1);
         case LEFT:
```

```
            return new Location(i, j - 1);
        case DOWN:
            return new Location(i + 1, j);
        case UP:
            return new Location(i - 1, j);
        default:
            throw new IllegalArgumentException();
    }
  }
}
```

```
public interface Board {
   /** Start the game. */
   void start();
   /** Gracefully terminate the program. */
   void stop();

   /** Move a Player from Location a to Location b…  */
   Result move(Location a, Location b);

   /** Return the lock for Location l.
    *  @throws ArrayIndexOutOfBoundsException if l is out of bounds of board. */
   Semaphore getLock(Location l);

    /** Return true iff the game has ended */
   boolean gameEnded();

   /** Return the number of columns (or rows) in the board. */
   int size();

   /** Return the distance to the next obstacle from Location a.
    *  An obstacle can be another Player, a wall (the Board bounds), or the goblet.
    *  If the game has ended, -1 is returned.
    *  @param a The location of the Player.
    *  @param dir The Direction to look at.
    *  @return distance to the obstacle.
    *  @throws IllegalArgumentException if a is out of bounds or if there is no Player
    *  at a. */
   int lookAhead(Location a, Direction dir);
}
```

גלעד הציע את המימוש הבא לממשק Board, שלכל תא במערך **משתמש ב- Semaphore עם permits=1**:

```
public final class GiladBoardImpl implements Board {
   private final Player[][] board;
   private final Semaphore[][] locks;
   private final Location goblet;
   …

   /** returns true iff l is within the bounds of the board */
   private boolean checkBoundsValidity(Location l) { … }

   private Player getPlayer(Location l) { return board[l.i][l.j]; }

   @Override
   public Semaphore getLock(Location l) { return locks[l.i][l.j]; }

   @Override
   public int lookAhead(Location a, Direction dir) throws InterruptedException {
      if (!checkBoundsValidity(a)) throw new IllegalArgumentException();
      if (gameEnded()) return -1;
      int distance = 0;
      Location current = a.move(dir);
      while (checkBoundsValidity(current)) {
         distance++;
         if (current.equals(goblet)) break;
         getLock(current).acquire();
         if (getPlayer(current) != null) break;
         current = current.move(dir);
      }

      int i = 1;
      current = a.move(dir);
      while (i < distance) {
         getLock(current).release();
         i++;
         current = current.move(dir);
      }
      return distance;
   }
}
```

א) הגדירו למתודה lookAhead תנאי התחלה וסיום (5 נקודות).

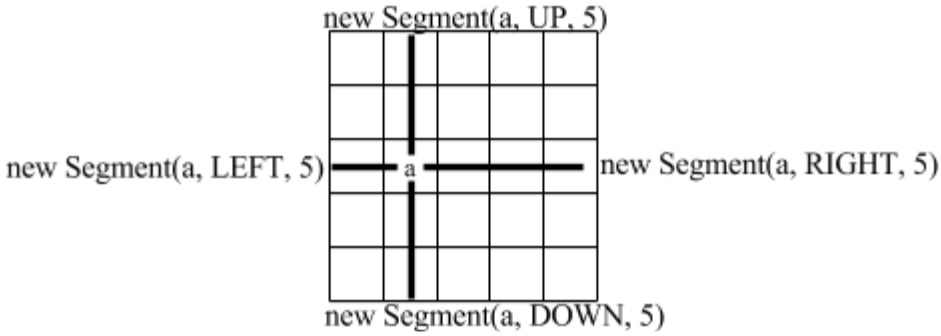
ב) יערית ציינה שיתכן חבק במימוש של גלעד למתודה lookAhead, הדגימו אותו (5 נקודות).

ג) כתבו מחדש את המתודה lookAhead ופתרו את החבק באמצעות Resource Ordering. (10 נקודות).
הנחיות:

- יש להשתמש במנעולים המוגדרים במערך lock, ואין להגדיר מנעולים נוספים.
- אין להוסיף מחלקות או לשנות קוד מחוץ למחלקה BoardImpl.
- פתרונות הנועלים את כל הלוח לא יתקבלו.

יערית הציעה מימוש אחר למתודה lookAhead, בו תופסים פחות מנעולים:
במקום להשתמש ב Semaphore אחד לכל תא, ולנעול תא-תא במסלול, נועלים אובייקט אחד (מסוג Segment) המייצג את כל התאים במסלול. באופן זה, אם ת'רד 1 תופס מנעול עבור Segment s1, ות'רד 2 מנסה לתפוס מנעול עבור s2, שחופף או חוצה את s1 - ת'רד 2 יחסם עד שיהיה ניתן לתפוס את s2.

בדיאגרמה הבאה ניתן לראות שקריאה לבנאי של Segment עם הפרמטרים a ו-Direction.LEFT ו-size()=5,
אזי ה-Segment שיחזור ייצג את התאים (2,1) ו-(2,0).



המחלקה Segment נתונה - ניתן להניח שכל המתודות שלה ממומשות (אופן מימושן אינו רלבנטי לשאלה):

```
public final class Segment {
   public final Location start, end;
   public Segment(Location start, Direction dir, int size) { … }

   @Override  public boolean equals(Object obj) { … }

   /** Returns true iff this Segment intersects other. */
   public boolean intersect(Segment other) { … }
}
```

להלן המימוש של יערית למחלקת הלוח, שימו לב לשינוי בממשק Board (המתודה getLock הוחלפה בשתי מתודות חדשות):

```
public interface Board {
   ...
   ~~Semaphore getLock(Location l);~~

   /** Acquires the lock for Segment s, blocking until it is possible to acquire it, or the
     *  thread is interrupted  */
   void acquireSegment(Segment s) throws InterruptedException;

   /** Release the Segment */
   void releaseSegment(Segment s);
}
```

```
public final class YaaritBoardImpl implements Board {
   private final List<Segment> acquiredSegments;
   …

   void acquireSegment(Segment s) throws InterruptedException {
       /** @TODO: COMPLETE (D) */
   }

   void releaseSegment(Segment s) {
       /** @TODO: COMPLETE (D) */
   }

   /** Returns the distance to the nearest Obstacle  */
   private int distanceToNearestObstacle(Location a, Direction dir) { … }

   public int lookAhead(Location a, Direction dir) {
      if (!checkBoundsValidity(a)) throw new IllegalArgumentException();
      if (gameEnded()) return -1;
      Segment segment = new Segment(a, dir, size());

      try {
          acquireSegment(segment);
          return distanceToNearestObstacle(a, dir);
       } finally {
          releaseSegment(segment);
       }
   }
}
```

ד) השלימו את המתודות acquireSegment ו-releaseSegment' כך שתפיסת ה Segment תהיה בטוחה לריצה מקבילית (כמובן שאין לנעול את כל הלוח). יש לשנות רק את המחלקה YaaritBoardImpl.
רמז: יש לפתור ע"י שימוש ב Version Iterator (פתרונות יצירתיים ויעילים אחרים יתקבלו).
(10 נקודות).

שאלה 2 (25 נקודות)

בקורס "משחקים בסי הכיף" הסטודנטים כתבו מימוש חלקי למשחק לוח.

א) להלן המחלקה Terrain אשר מנהלת את לוח המשחק על ידי מערך דו ממדי בגודל (width_ X height_) תאים.כל תא מכיל איבר מסוג T. ממשו את שני הבנאים והפונקציה הורסת אשר הוגדרו במחלקה זו. עבור בנאי מעתיק, ממשו deep copy. (8 נקודות)

```
template<typename T> class Terrain{
public:
        Terrain(const Terrain& other);
        Terrain(int n, int m);
        ~Terrain();

        T&      getCell(int x, int y) { return array_[x][y]; }
        void    setCell(int x, int y, T t) { array_[x][y] = t; }
private:
        T** array_;
        int width_;
        int height_;
};
```

ב) נתון כי במשחק שני סוגי כלים, (Knight, Cavalier). בהמשך, הסטודנטים מימשו עוד חלקים של המשחק כדלקמן:

```
class GameFigure{
public:
        virtual void move() = 0;
};

class Knight : public GameFigure{
public:
        virtual void move(){...}//הנח מימוש קיים
};
class Cavalier : public GameFigure{
public:
        virtual void move() {...}//הנח מימוש קיים
};

template<typename T> class Game{
public:
        Game(const Game &other) :terrain_(other.terrain_) {}
        void Init(int n, int m, T *gamefigures_array);
private:
```

```
        Terrain<T>  terrain_;
};
void main(){
        int n = 10, m = 10;
        //***put your code here***//
        game1.Init(n, m, array);
}
```

עליכם להשלים את הקוד בmain על מנת שהמשחק יוכל לעבור אתחול בהצלחה.שימו לב לדרישות: game1 הוא אובייקט מסוג Game על הstack . array הנו מערך (חד ממדי) בן 100 מצביעים בheap אשר מחזיק בתאים זוגיים אובייקטים מסוג Cavalier ובתאים אי-זוגיים אובייקטים מסוג Knight. הניחו כי קיים מימוש של Init.
(9 נקודות)

ג) נוספו מימוש של Factory במחלקה Terrain, וכן המימוש של Init במחלקה Game:

```
template<typename T> class Terrain{
public:
...
static Terrain  generateTerrain(int n, int m) { Terrain *t = new Terrain(n, m); return *t; }
…
}

template<typename T> void Game<T>::Init(int n, int m, T *gamefigures_array){

        terrain_ = Terrain<T>::generateTerrain(n, m); //@@

        int index = 0;
        for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                        terrain_.setCell(i, j, gamefigures_array[index++]);
}
```

הוסיפו את הנדרש (בלבד) למחלקה Terrain על מנת שהשורה המסומנת ב //@@ תתבצע באופן יעיל (רמז rule of 5).
(8 נקודות)

שאלה 3	(30 נקודות)

בשאלה נשתמש בתבנית העיצוב Command Invocation Protocol (הקוד, כפי שנלמד בכיתה, מופיע בנספח המבחן), כדי לממש אפליקציה בסיסית לבנק.

באפליקציה זו, הממשק בין הלקוח לשרת מבוסס על פרוטוקול בן שלוש פקודות:

LOGIN name

הלקוח מבקש מהשרת להצטרף תחת השם **name**, השרת מצרף את הלקוח ומחזיר אישור **ack**

DEPOSIT money

הלקוח מבקש מבקש להפקיד בחשבונו סכום כסף (**money**), השרת מגדיל את סכום חשבון הלקוח בהתאם ומחזיר ללקוח אישור **ack**

Transfer money name

הלקוח מבקש להעביר סכום כסף (**money**) ללקוח אחר (**name**), השרת מחזיר אישור חיובי **ack** אם היה מספיק כסף בחשבון, ותשובה שלילית **nak** אם ללא היה מספיק כסף בחשבון. אם היה מספיק כסף, הסכום מועבר ע"י השרת לחשבון של **name**, אחרת הכסף אינו מועבר. אין אפשרות למשיכת יתר.

א. הגדירו את מבנה הנתונים **BankState** הנדרש בשרת כדי לתמוך בפרוקטול זה (7 נקודות)

```
public class BankState {
   //data structure to support the Bank protocol
   // @TODO
}
```

ב. השלימו את המחלקות **LoginCommand, DepositCommand, TransferCommand** המממשות את שלוש הפקודות של הפרוטוקול (9 נקודות)

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

public class DepositCommand implements Command<BankState> {
   //implementation of the deposit command
   // @TODO
}

public class TransferCommand implements Command<BankState> {
   //implementation of the transfer command
   // @TODO
}
```

ג. השלימו את התוכנית הראשית של השרת (6 נקודות)

```
public class BankServer {
   public static void main(String[] args) {
            // @TODO
   }
}
```

ד. השלימו את התוכנית **BankClients**, המריצה במקביל שני לקוחות - חילַק ובילַק -, כאשר כל לקוח מבצע את הפעולות הבאות:

<u>חילק</u>:
- מבצע **login**
- מפקיד 1000 ש"ח
- מכאן אילך, מעביר שוב ושוב 1000 ש"ח לבילק

<u>בילק</u>:
- מבצע **login**
- מפקיד 1000 ש"ח
- מכאן אילך, מעביר שוב ושוב 1000 ש"ח לחילק

במידה ופעולה לא הצליחה, הלקוח ידפיס זאת על המסך.

```
public class BankClients {
  public static void main(String[] args) {
        // @TODO
  }
}
```

(8 נקודות)

שאלה 4 (15 נקודות)

בשאלה זו עליכם לייצר persistence layer למערכת לניהול מוסך. ה-persistence layer יתמוך בצרכים הבאים:

- הכנסת עבודה חדשה: לכל עבודה יש מספר רכב, תיאור תקלה ומספר עבודה.
- רישום עובד חדש: העובד ימלא את הפרטים הבאים: שם ות"ז.
- שיוך עובד לעבודה: כאשר מנהל המערכת רוצה לשייך עבודה לעובד הוא ימלא את הפרטים הבאים: ת"ז עובד, מספר עבודה, בנוסף, הוא מציין שהעבודה עדיין לא הסתיימה.
- סיום עבודה: כאשר עובד מסיים את אחת העבודות המוקצות לו - הוא מדווח על מספר העבודה ומאותו הרגע היא נחשבת גמורה.
- בירור עבודות בטיפול: בעל המוסך יכול לבקש להדפיס את כל מספרי העבודות שעדיין לא הסתיימו

א) הסיקו אלו טבלאות צריכות להיות קיימות במסד הנתונים כדי לתמוך בדרישות המוסך. כתבו את הפקודות הדרושות כדי לייצר את הטבלאות (פקודות CREATE TABLE...)יש להקפיד על מפתחות ראשיים וזרים. (5 נק')

הבהרות:

- בעת הכנסת עבודה ניתן להניח שמנהל המוסך הוא זה שיספק את מספר העבודה שישמר במסד הנתונים.

ב) רשמו בפייתון את מחלקות הDTO הדרושות לפי הטבלאות שהגדרתם. (5 נק')

ג) רשמו לכל מחלקת DTO, מחלקת DAO מתאימה (5 נק')

הבהרות:

- מחלקות הDAO יקבלו בבנאי אוביקט מסוג connection של sqllite3.
- אין להשתמש במחלקת ה-DAO הגנרית אותה למדנו בכיתה, יש לייצר בעצמכם פונקציות כגון insert, find_all, update וכו'.
- בכל מחלקת DAO, אין צורך לייצר פעולות אשר המוסך, לפי הצרכים הנתונים מעלה לא דורש (לדוגמא, עבור שום DAO אין צורך לייצר פעולות של delete מכיוון שאין אף דרישה למחוק נתונים לפי צרכי המוסך, בנוסף יש מחלקות שלא דורשות find_all וכו')
- בפונקציות השונות אין צורך לדאוג לתקינות קלט או לטיפול בשגיאות

לנוחיותכם מצורפת תבנית הקוד הבאה (שמות המחלקות והפונקציות בתבנית הם לצורך הדוגמא בלבד)

```
def create_tables(conn):
   conn.execute("""
         CREATE TABLE ...
   """)

#Data Transfer Objects
class Dto1(object):
  def __init__(self, ...):

...

#Data Access Objects
```

```
class Dao1(object):
    def __init__(self, conn):
            self._conn = conn

    def insert(self, dto1):
        self._conn.execute("""
            INSERT INTO ...
        """)

    def find_all(self):
      c = self._conn.cursor()
      c.execute("""
        SELECT ... FROM ...
      """)
      return [Dto1(*row) for row in c.fetchall()]
    ...
...
```

נספח

א. תבנית ה Command Invocation Protocol, כפי שנלמדה בכיתה

```
public interface MessageEncoderDecoder<T> {
	T decodeNextByte(byte nextByte);
	byte[] encode(T message);
}

public interface MessagingProtocol<T> {
	T process(T msg);
	boolean shouldTerminate();
}


public interface Command<T> extends Serializable {

	Serializable execute(T arg);
}


public class RemoteCommandInvocationProtocol<T> implements MessagingProtocol<Serializable> {

	private T arg;

	public RemoteCommandInvocationProtocol(T arg) {
		this.arg = arg;
	}

	@Override
	public Serializable process(Serializable msg) {
		return ((Command) msg).execute(arg);
	}

	@Override
	public boolean shouldTerminate() {
		return false;
	}

}


public class ObjectEncoderDecoder<> implements MessageEncoderDecoder<Serializable> {

	private final byte[] lengthBytes = new byte[4];
	private int lengthBytesIndex = 0;
	private byte[] objectBytes = null;
	private int objectBytesIndex = 0;

	@Override
	public Serializable decodeNextByte(byte nextByte) {

	   if (objectBytes == null) { //indicates that we are still reading the length
	      lengthBytes[lengthBytesIndex++] = nextByte;
	      if (lengthBytesIndex == lengthBytes.length) {
	         //we read 4 bytes and therefore can take the length
	         int len = bytesToInt(lengthBytes);
	         objectBytes = new byte[len];
	         objectBytesIndex = 0;
	         lengthBytesIndex = 0;
	      }
	   } else {
```

```
            objectBytes[objectBytesIndex] = nextByte;
            if (++objectBytesIndex == objectBytes.length) {
                Serializable result = deserializeObject();
                objectBytes = null;
                return result;
            }
        }
        return null;
    }

    private static void intToBytes(int i, byte[] b) {
        b[0] = (byte) (i >> 24);
        b[1] = (byte) (i >> 16);
        b[2] = (byte) (i >> 8);
        b[3] = (byte) i;
    }

    private static int bytesToInt(byte[] b) {
        //this is the reverse of intToBytes,
        //note that for every byte, when casting it to int,
        //it may include some changes to the sign bit so we remove those by anding with 0xff

        return ((b[0] & 0xff) << 24)
                | ((b[1] & 0xff) << 16)
                | ((b[2] & 0xff) << 8)
                | (b[3] & 0xff);
    }

    @Override
    public byte[] encode(Serializable message) {
        return serializeObject(message);
    }

    private Serializable deserializeObject() {
        try {
            ObjectInput in = new ObjectInputStream(new ByteArrayInputStream(objectBytes));
            return (Serializable) in.readObject();
        } catch (Exception ex) {
            throw new IllegalArgumentException("cannot desrialize object", ex);
        }
    }

    private byte[] serializeObject(Serializable message) {
        try {
            ByteArrayOutputStream bytes = new ByteArrayOutputStream();

            //placeholder for the object size
            for (int i = 0; i < 4; i++)
                bytes.write(0);

            ObjectOutput out = new ObjectOutputStream(bytes);
            out.writeObject(message);
            out.flush();
            byte[] result = bytes.toByteArray();

            //now write the object size
            intToBytes(result.length - 4, result);
            return result;

        } catch (Exception ex) {
            throw new IllegalArgumentException("cannot serialize object", ex);
        }
    }

}
```

```
public class RCIClient implements Closeable {

        private final ObjectEncoderDecoder encdec;
        private final Socket sock;
        private final BufferedInputStream in;
        private final BufferedOutputStream out;

        public RCIClient(String host, int port) throws IOException {
           sock = new Socket(host, port);
           encdec = new ObjectEncoderDecoder();
           in = new BufferedInputStream(sock.getInputStream());
           out = new BufferedOutputStream(sock.getOutputStream());
        }


        public void send(Command<?> cmd) throws IOException {
           out.write(encdec.encode(cmd));
           out.flush();
        }

        public Serializable receive() throws IOException {
           int read;
           while ((read = in.read()) >= 0) {
              Serializable msg = encdec.decodeNextByte((byte) read);
              if (msg != null)
                 return msg;
            }
            throw new IOException("disconnected before complete reading message");
        }

        @Override
        public void close() throws IOException {
           out.close();
           in.close();
           sock.close();
        }

}
```