תאריך הבחינה: 08.03.2022

שם המרצים: פרופ אנדרי שרף

ד"ר מני אדלר

ד"ר מרינה קוגן-סדצקי

שם הקורס: תכנות מערכות

מספר הקורס: 202-1-2031

שנה: 2022 סמסטר: א' מועד: ב

משך הבחינה: שעתיים וחצי

חומר עזר: אין

- במבחן זה מספר שאלות, הניקוד שלהן לא בהכרח אחיד
- שימו לב: במבחן זה אין אופציה לכתוב "לא יודע/ת" , זה לא מזכה בניקוד
- השאלות הן רבי ברירה. יש לבחור רק תשובה אחת שהיא הכי נכונה
- יש לסמן בצורה ברורה את התשובות

בהצלחה !

המחלקה IntVectorUtils מכילה גרסאות שונות לפעולה smooth המקבלת ווקטור של מספרים שלמים ומחזירה ווקטור המכיל את האברים במקומות הזוגיים (במקום 0,2,4 וכו') של הווקטור המקורי.
לדוגמא, עבור הווקטור {0,1,2,3,4,5} פונקציות ה-smooth השונות מחזירות את הווקטור {0,2,4}.
הגרסאות השונות של smooth נבדלות רק בדרך העברת הווקטור ו/או באופן החזרתו.
הפונקציה printVector מדפיסה את תכולת הווקטור.
כמו כן מוגדר המשתנה v כווקטור עם המספרים {0,1,2,3}.

```
#include <vector>
typedef std::vector<int> IntVector;

IntVector smooth1(const IntVector& v)        {
   IntVector tmp;
   for (int i = 0; i < v.size(); i+=2)
      tmp.push_back(v.at(i));
    return tmp;
}

IntVector& smooth2(const IntVector& v) {
   IntVector tmp;
   for (int i = 0; i < v.size(); i+=2)
      tmp.push_back(v.at(i));
    return tmp;
}

IntVector* smooth3(const IntVector& v) {
   IntVector* tmp = new IntVector();
   for (int i = 0; i < v.size(); i+=2)
      tmp->push_back(v.at(i));
    return tmp;
}

void smooth4(IntVector* v) {
   IntVector* tmp = new IntVector();
   for (int i = 0; i < v->size(); i+=2)
      tmp->push_back(v->at(i));
  delete v;
  v = tmp;
}

void smooth5(IntVector*& v) {
   IntVector* tmp = new IntVector();
   for (int i = 0; i < v->size(); i+=2)
      tmp->push_back(v->at(i));
```

```
  delete v;
  v = tmp;
 }

 void printVector(const IntVector& v) {
    for (int i=0; i<v.size(); i++) {
       if (i>0) std::cout << ",";
       std::cout << v.at(i);
    }
    std::cout << '\n';
}

void printVector(IntVector* v) {
   printVector(*v);
}

IntVector* v = new IntVector();
v->push_back(0);
v->push_back(1);
v->push_back(2);
v->push_back(3);
```

1. **מה הפלט של קטע הקוד הבא [4 נקודות]**

```
IntVector v1(smooth1(*v));
printVector(v1);
```

א. 0,2
ב. פניה למקום לא מוגדר במחסנית
ג. פניה למקום מחוק ב heap
ד. 0,1,2,3
ה. אף תשובה אינה נכונה

2. **מה הפלט של קטע הקוד הבא [4 נקודות]**

```
IntVector& v2 = smooth2(*v);
printVector(v2);
```

א. 0,2
ב. פניה למקום לא מוגדר במחסנית
ג. פניה למקום מחוק ב heap
ד. 0,1,2,3
ה. אף תשובה אינה נכונה

3. מה הפלט של קטע הקוד הבא [4 נקודות]

```
IntVector* v3 = smooth3(*v);
printVector(v3);
```

א. פניה למקום לא מוגדר במחסנית
ב. פניה למקום מחוק ב heap
ג. 0,2
ד. 0,1,2,3
ה. אף תשובה אינה נכונה

4. מה הפלט של קטע הקוד הבא [4 נקודות]

```
smooth4(v);
printVector(v);
```

א. פניה למקום לא מוגדר במחסנית
ב. 0,2
ג. 0,1,2,3
ד. פניה למקום מחוק ב heap
ה. אף תשובה אינה נכונה

5. מה הפלט של קטע הקוד הבא [4 נקודות]

```
smooth5(v);
printVector(v);
```

א. 0,2
ב. פניה למקום מחוק ב heap
ג. פניה למקום לא מוגדר במחסנית
ד. 0,1,2,3
ה. אף תשובה אינה נכונה

6. איזו גירסה של smooth היא היעילה ביותר – כלומר, איזו גירסה עובדת ללא שגיאות תוך העתקה מינימאלית של מידע? [5 נקודות]

א. smooth1
ב. smooth2 ו smooth4
ג. smooth1 אך ורק אם ממומשות במחלקה vector המתודות move 'move constructor copy assignment operator
ד. Smooth2
ה. smooth3 ו smooth5

7. המחלקה Even מייצגת מספר זוגי [7 נקודות]

```
final class Even {
    //@INV: get() %2 == 0
    private volatile int val;
    Even(int v) throws Exception {
        if (v %2 != 0)
           throw new Exception("Odd number!");
        val = v;
    }
    public int get() { return val; }
    public Even next() { return new Even(val +2);
}
```

סמנו את התשובה הנכונה:

א. המחלקה Even בטוחה תחת כל חישוב מקבילי, וכן השימוש בה בהכרח מקיים נכונות
ב. המחלקה Even אינה בטוחה תחת כל חישוב מקבילי, אך השימוש בה בהכרח מקיים נכונות
ג.המחלקה Even בטוחה תחת כל חישוב מקבילי, אך שימוש בה אינו מקיים בהכרח נכונות
ד. אף תשובה אינה נכונה

8. נתונה המחלקה Synchronizer המסדירה את הכניסה לקטעי קוד קריטיים עבור ת'רדים מסוגים שונים [8 נקודות]

```
class Synchronizer {

    enum ThreadType { USER, MANAGER, WORKER };

   private Map<ThreadType, Integer> mapThreadType2Count;

   Synchronizer() {
        mapThreadType2Count = new HashMap<ThreadType, Integer>();
        mapThreadType2Count.put(ThreadType.USER,0);
        mapThreadType2Count.put(ThreadType.MANAGER,0);
        mapThreadType2Count.put(ThreadType.WORKER,0);
   }

    protected synchronized void before(ThreadType tt) {
        while(!allow(tt))
            wait();
```

```
        mapThreadType2Count.set(tt,mapThreadType2Count.get(tt)+1);
    }
    protected synchronized void after(ThreadType tt) {
        mapThreadType2Count.set(tt,mapThreadType2Count.get(tt)-1);
        notifyAll();
    }

    protected boolean allow(ThreadType tt) {
        return (tt == ThreadType.MANAGER &&
                 mapThreadType2Count.get(ThreadType.WORKER) == 0)
                ||
                (tt == ThreadType.USER && mapThreadType2Count.get(ThreadType.USER) <=
                                        mapThreadType2Count.get(ThreadType.WORKER))
                ||
                (tt == ThreadType.WORKER &&
                 (mapThreadType2Count.get(ThreadType.MANAGER) == 0 &&
                                        mapThreadType2Count.get(ThreadType.USER) ==
                                         mapThreadType2Count.get(ThreadType.WORKER)));
    }
}
```

נתונה מערכת ובה ת'רדים מסוגים שונים - USER, ACTION, MANAGER (ייתכנו כמה ת'רדים מכל סוג). נתון כי כל אחד מהת'רדים במערכת מבצע before לפני כניסתו לקטע קוד קריטי ו after בסוף קטע הקוד הקריטי, עם הסוג שלו כפרמטר.

בדיון שנערך בכיתה הועלו התובנות הבאות ביחס לפעולות before ו after של הת'רדים השונים:

1. **תיתכן הרעבה של ת'רדים מכל סוג שהוא**
2. **ייתכן שת'רד מסוג USER ות'רד מסוג MANAGER יהיו בקטע קוד קריטי**
3. **ייתכן שת'רד מסוג USER ות'רד מסוג WORKER יהיו בקטע קוד קריטי**
4. **ייתכן תרחיש של LiveLock**

סמנו את התשובה הנכונה:

א. קביעות 1,3 נכונות
ב. קביעות 1,2,3 נכונות
ג. קביעות 1,4,3 נכונות
ד. קביעות 2,4 נכונות
ה. כל הקביעות נכונות
ו. אף קביעה נכונה

9. **נתונים שני מימושים עבור מחלקת ה blocking-queue (לשם פשטות נתעלם מתרחישי InterruptedException) [8 נקודות]**

```
class WaitQueue<T> {
   private List<T> lst;
   int capacity;

  WaitQueue(int capacity) throws Exception {
       if (capacity < 1) throw new Exception("empty queue");
       this.capacity = capacity;
       lst = new LinkedList<T>();
  }

   public synchronized T remove() {
        while (lst.size() == 0)
             wait();
        T ret =  lst.remove(lst.size()-1);
        notifyAll();
        return ret;
   }


    public synchronized void add(T obj) {
        while (lst.size() == capacity)
           wait();
        lst.add(obj);
        notifyAll();
   }
}


class SleepQueue<T> {
    private List<T> lst;
    int capacity;
    Semaphore sem;

   SleepQueue(int capacity) throws Exception {
        if (capacity < 1) throw new Exception("empty queue");
        this.capacity = capacity;
        lst = new LinkedList<T>();
        sem = new Semaphore(1,true); // FairSemaphore
   }

    public T remove() {
         sem.acquire();
         while (lst.size() == 0) {
              sem.release();
              Thread.currentThread().sleep(1000);
              sem.acquire();
         }
         T ret = lst.remove(lst.size()-1);
```

```
        sem.release();
        return ret;
    }


    public void add(T obj) {
        sem.acquire();
        while (lst.size() == capacity) {
            sem.release();
            Thread.currentThread().sleep(1000);
            sem.acquire();
        }
        lst.add(obj);
        sem.release();
    }
}
```

נתונים שני <u>סוגי</u> ת'רדים, האחד מוסיף אובייקטים לתור משותף על ידי הקריאה למתודה add, והשני לוקח אובייקטים מהתור המשותף ע"י הקריאה למתודה remove (תבנית producer-consumer)

בדיון בכיתה הועלו הקביעות הבאות:

1. **בתור משותף מטיפוס WaitQueue, כל אובייקט שהוכנס ע"י ת'רד אחד, יילקח ע"י הת'רד השני**
2. **בתור משותף מטיפוס SleepQueue, כל אובייקט שהוכנס ע"י ת'רד אחד, יילקח ע"י הת'רד השני**
3. **מנגנון הסנכרון של SleepQueue הוגן יותר**
4. **מנגנון ההמתנה של WaitQueue יעיל יותר**

ציינו את התשובה הנכונה:

א. קביעות 3,4 נכונות
ב. קביעות 1,2 נכונות
ג. קביעות 1,2,3 נכונות
ד. קביעות 2,3,4 נכונות
ה. כל הקביעות נכונות
ו. אף קביעה אינה נכונה

10. נתון כי בחבילה החדשה של Java נוספה למחלקה Thread גרסה נוספת למתודה sleep: קריאה ל ()sleep ללא פרמטר מעבירה את הת'רד בכל מקרה למצב blocked עד אשר יבוצע עליו ()7] interrupt נקודות]

המימוש הבא של blocking-queue עושה שימוש בגרסה הנוספת של מתודת ה-sleep באופן הבא (השדה otherThreads מאותחל עם כל שאר הת'רדים במערכת, ניתן להניח שלא נוספים אחר כך ת'רדים נוספים):

```
class SleepInterruptQueue<T> {
    private List<T> lst;
    private List<Thread> otherThreads;
    int capacity;
    Semaphore sem;
```

```
SleepInterruptQueue(int capacityת List<Thread> otherThreads) throws Exception {
    if (capacity < 1) throw new Exception("empty queue");
    this.capacity = capacity;
    lst = new LinkedList<T>();
    sem = new Semaphore(1,true); // FairSemaphore
    this.otherThreads = otherThreads;
}

public T remove() {
    sem.acquire();
    while (lst.size() == 0) {
        sem.release();
        try { Thread.currentThread().sleep(); } catch (InterruptedException) {}
        sem.acquire();
    }
    T ret = lst.remove(lst.size()-1);
    for (Thread t : otherThreads)
        t.interrupt();
    sem.release();
    return ret;
}


public void add(T obj) {
    sem.acquire();
    while (lst.size() == capacity) {
        sem.release();
        try { Thread.currentThread().sleep(); } catch (InterruptedException) {}
        sem.acquire();
    }
    lst.add(obj);
    for (Thread t : otherThreads)
        t.interrupt();
    sem.release();
}
}
```

נתונים שני ת'רדים, האחד מוסיף אובייקטים לתור משותף מטיפוס SleepInterruptQueue על ידי הקריאה למתודה add, והשני לוקח אובייקטים מהתור המשותף ע"י הקריאה למתודה remove (תבנית producer-consumer)

בדיון בכיתה הועלו הקביעות הבאות:

1. **ייתכן שת'רד יישן לנצח על אף שמתקיים תנאי ההתחלה עבור הפעולה שהוא מעוניין לבצע**
2. **כל אובייקט שהוכנס ע"י ת'רד אחד בהכרח יילקח ע"י הת'רד השני**
3. **מנגנון הסנכרון של SleepInterruptQueue הוגן יותר ממנגנון ההמתנה של SleepQueue (בשאלה הקודמת)**

4. **מנגנון ההמתנה של SleepInterruptQueue יעיל יותר ממנגנון ההמתנה של SleepQueue (בשאלה הקודמת)**

ציינו את התשובה הנכונה:

א. קביעה 2 נכונה
ב. קביעות 2,3 נכונות
ג. קביעות 1,4 נכונות
ד. קביעות 3,4 נכונות
ה. קביעה 4 נכונה
ו. אף קביעה אינה נכונה
ז. כל הקביעות נכונות

11. בדיון בכיתה הועלו הקביעות הבאות ביחס לשרת בתבנית ThreadPerClient המבוסס על פרוטוקול התקשורת UDP: [8 נקודות]

1. **לא כל ההודעות שהלקוח שולח יגיעו בהכרח לשרת**
2. **השרת עשוי לטפל בהודעות של הלקוח שלא על פי הסדר שבו הן נשלחו**
3. **מספר ה sockets בשרת צריך להיות כמספר הלקוחות**
4. **השרת יכול לשלוח בפעולה אחת הודעות למספר לקוחות**
5. **ממשק ה MessageEncoderDecoder בשרת נשאר ללא שינוי**
6. **ממשק ה MessagingProtocol בשרת נשאר ללא שינוי**

סמנו את התשובה הנכונה:

א. כל הקביעות נכונות
ב. קביעות 1,2,3,4 נכונות
ג. קביעות 1,2,4,6 נכונות
ד. קביעות 4,5,6 נכונות
ה. קביעות 1,2,3 נכונות
ו. קביעות 1,2,5,6 נכונות

12. אחד הסטודנטים בקורס הציע לשנות את המתודה complete במחלקת ה-ActorThreadPool באופן הבא (קוד הריאקטור מופיע בנספח): [8 נקודות]

```
private void complete(T act) {
    Queue<Runnable> pending = pendingRunnablesOf(act);
    if (pending.isEmpty())
        playingNow.remove(act);
    else
        execute(pending.poll(), act);
        try { pending.poll().run(); } finally { complete(act); }
```

```
}
```

בעקבות השינוי המוצע, אילו מבין התרחישים הבאים <u>עשוי</u> להתקיים:

1. **הרעבה של לקוח כלשהו**
2. **חוסר הוגנות בחלוקת זמן השרת לטיפול בלקוחות שונים**
3. **שני ת'רדים ב executor מטפלים באותו לקוח**
4. **אי שמירה על סדר ביצוע ההודעות של אותו לקוח**

א. תרחישים 1,3
ב. תרחישים 1,2
ג. תרחישים 2,3
ד. תרחישים 3,4
ה. תרחישים 2,4
ו. אף תרחיש
ז. כל התרחישים

13. בקורס SPL, אחד הסטודנטים הציע להשתמש ב ExecutorService רגיל במקום ה ActorThreadPool כפי שנלמד בכיתה(קוד הריאקטור מופיע בנספח). אם נקבל את דעתו, אילו מהקביעות הבאות נכונות: [7 נקודות]

1. **עשויה להתקבל הודעה עם הרכב בתים שהלקוח לא שלח**
2. **ההודעות לא יבוצעו בהכרח בסדר שבו הן נשלחו**
3. **ייתכן ששני ת'רדים ב executor יבצעו במקביל הודעות שונות של אותו לקוח**

א. קביעות 1,2
ב. קביעות 2,3
ג. קביעות 1,3
ד. קביעה 1
ה. קביעה 2
ו. קביעה 3
ז. כל הקביעות נכונות

חומר עזר: משימת ה-run של הת'רדים ב-executor

```
public class NonBlockingConnectionHandler {
  ...
  public Runnable continueRead() {
        ...
        return () -> {  // Return a task for the executor, based on the bytes read from the client
              while (buf.hasRemaining()) {
                 T nextMessage = encdec.decodeNextByte(buf.get());
                 if (nextMessage != null) {
```

```
                T response = protocol.process(nextMessage);
                if (response != null) {
                    writeQueue.add(ByteBuffer.wrap(encdec.encode(response)));
                    reactor.updateInterestedOps(chan,
                                SelectionKey.OP_READ | SelectionKey.OP_WRITE);
                }
            }
        }
    };
  }
  ...
}
```

14. בהמשך, אחד הסטודנטים בקורס הציע להשתמש ב ExecutorService רגיל במקום ה ActorThreadPool, תוך סנכרון משימת ה run של הת'רדים ב executor באופן הבא (קוד הריאקטור מופיע בנספח): [7 נקודות]

```
public class NonBlockingConnectionHandler {
  ...
  public Runnable continueRead() {
        ...
        return () -> {  // 2. Return a task for the executor, based on the bytes read from the client
          synchronized (NonBlockingConnectionHandler.this) {
            while (buf.hasRemaining()) {
                T nextMessage = encdec.decodeNextByte(buf.get());
                if (nextMessage != null) {
                    T response = protocol.process(nextMessage);
                    if (response != null) {
                        writeQueue.add(ByteBuffer.wrap(encdec.encode(response)));
                        reactor.updateInterestedOps(chan,
                                    SelectionKey.OP_READ | SelectionKey.OP_WRITE);
                    }
                }
            }
          }
        };
  }
  ...
}
```

אם נקבל את דעתו, אילו מהקביעות הבאות נכונות:

1. **עשויה להתקבל הודעה עם הרכב בתים שהלקוח לא שלח**
2. **ההודעות לא יבוצעו בהכרח בסדר שבו הן נשלחו**
3. **ייתכן ששני ת'רדים ב executor יבצעו במקביל הודעות שונות של אותו לקוח**

א. קביעה 3
ב. כל הקביעות
ג. קביעות 2,3
ד. קביעות 1,3
ה. קביעה 1
ו. קביעות 1,2
ז. קביעה 2

בקורס תכנות מערכות רוצים לבנות persistence layer המתאר את שני המועדים של המבחנים בסמסטר הנוכחי. צריך לתמוך במידע הבא:

לכל שאלה ממבחנים של תכנות מערכות יש :

- **מספר מזהה ייחודי (id)**
- **שייכות למועד (א או ב)**
- **נושא השאלה ('Memory', 'Concurrency', 'Network','DataBase')**
- **מספר הסטודנטים שענו נכון על השאלה**
- **רמת השאלה (קלה = 1, בינונית = 2, קשה = 3)**

לכל סטודנט יש :

- **ת.ז. ייחודי (מספר חיובי שלם בן 9 ספרות)**
- **רשימת הקורסים הנלמדים (לפי id של הקורסים)**
- **עבור כל קורס**
  - **במידה וסטודנט ניגש למועד א, ציון של מועד א**
  - **במידה וסטודנט ניגש למועד ב, ציון של מועד ב**
- **ממוצע של ציונים עד עכשיו בתואר**

לכל קורס יש:

- **מספר הקורס ייחודי (id)**
- **שם הקורס ('SPL', 'Archi', 'OS')**
- **ממוצע של המבחנים בקורס על פני כל השנים הקודמות (לא כולל את השנה הנוכחית)**

15. השלימו שורות עם סימני שאלה (או השאירו ריקות אם אין צורך בהשלמה) SQL ליצירת טבלאות הנדרשות. אין לייחס חשיבות לפסיקים בגוף השאילתות. [5 נקודות]

```
def create_tables():
connection.execute("""
    CREATE TABLE SPL_exams_questions (
        ???
        moed TEXT  NOT NULL,
```

```
6.          topic TEXT  NOT NULL,
7.          numOfCorrectAnswers INT  NOT NULL,
8.          level INT  NOT NULL,
9.         ???
10.         ???
11. )""")
12. connection.execute("""
13.     CREATE TABLE students (
14.        id INT PRIMARY KEY,
15.        average INT NOT NULL,
16. )""")
17. connection.execute("""
18.     CREATE TABLE grades (
19.        ???
20.        ???
21.        ???
22.        ???
23.        FOREIGN KEY(sID) REFERENCES students(id),
24.        FOREIGN KEY(courseID) REFERENCES courses(id),
25.        ???
26. )""")
27. connection.execute("""
28.     CREATE TABLE courses (
29.        id INT PRIMARY KEY,
30.        name TEXT NOT NULL,
31.        average INT NOT NULL,
32. )""")
```

א.
(4) id INT PRIMARY KEY
(19) sID INT NOT NULL
(20) courseID INT NOT NULL
(21) moed TEXT NOT NULL
(22) grade INT NOT NULL
(25) PRIMARY KEY (sID, courseID, moed)

ב.
(4) id INT PRIMARY KEY
(9) FOREIGN KEY(moed) REFERENCES grades(moed)
(19) sID INT PRIMARY KEY
(20) courseID INT PRIMARY KEY
(21) moed TEXT PRIMARY KEY
(22) grade PRIMARY KEY

ג.
(4) id INT
(9) PRIMARY KEY (id, moed)

(19) sID INT PRIMARY KEY
(20) courseID INT PRIMARY KEY
(21) moed TEXT NOT NULL
(22) grade INT NOT NULL

ד.
(4) id INT
(9) FOREIGN KEY(moed) REFERENCES grades(moed)
(10) PRIMARY KEY (id, moed)
(19) sID INT NOT NULL
(20) courseID INT NOT NULL
(21) moed TEXT NOT NULL
(22) grade INT NOT NULL
(25) PRIMARY KEY (sID, courseID)

16. לאחר המבחנים, צוות הקורס תכנות מערכות (SPL) רוצה לחשב סטטיסטיקה על המבחנים של השנה. יש לחשב ממוצע של מועד (הנתון כארגומנט לפונקציה) של קורס תכנות מערכות של השנה (עבור הסטודנטים שניגשו אליו). השלימו את החסר. [5 נקודות]

```
1. def average(moed):
2.     cursor = connection.cursor()
3.     all = cursor.execute("""
4.    SELECT grades.grade
5.    ???
6.    ???
7.    ???
8.    """, [moed]).fetchall()
9.     n = len(all)
10.        ???
11.        sum = reduce(lambda x,y: x+y, all)
12.        return sum/n
print("average SPL moed A:" , average('A'))
print("average SPL moed B:" , average('B'))
```

א.
(5) FROM grades JOIN courses
(6) ON grades.courseID = courses.id
(7) WHERE courses.name = 'SPL' and grades.moed = ?

(10) all = [item[0] for item in all]

ב.
(5) FROM grades
(6) WHERE grades.moed = ?
(10) all = [item[0] for item in all]

ג.
(5) FROM grades JOIN courses
(6) ON grades.courseID = courses.id
(7) WHERE courses.name = 'SPL' and grades.moed = ?

ד.
(5) FROM grades
(6) WHERE grades.moed = ?

17. בהנתן רמה (level) ומועד (moed) (הנתונים כארגומנטים לפונקציה), חישבו % סטודנטים שענו נכון על השאלות של הרמה הזו במועד המבוקש של מבחן SPL. [5 נקודות]

```
1. def succeeded(level, moed):
2.    cursor = connection.cursor()
3.    all = cursor.execute("""
4.      ???
5.      FROM SPL_exams_questions
6.      WHERE level = ? and moed = ?
7.      """, [level, moed]).fetchall()
8.    numOfQuestions = len(all)
9.    all = [item[0] for item in all]
10.    sum = reduce(lambda x,y: x+y, all)
11.    all = cursor.execute("""
12.      SELECT sID
13.   ???
14.   ???
15.   ???
16.    """,[moed]).fetchall()
17.    n = len(all)
18.    return sum*100/(n*numOfQuestions)

print("% succeeded:" , succeeded(3,'A')) # 3 is hard
print("% succeeded:" , succeeded(3,'B')) # 3 is hard
```

א.
(4) SELECT numOfCorrectAnswers
(13) FROM grades JOIN courses

(14) ON courses.id = grades.courseID
(15) WHERE courses.name = 'SPL' and grades.moed = ?

ב.
(4) SELECT moed, numOfCorrectAnswers
(13) FROM grades JOIN courses JOIN SPL_exams_questions
(14) ON courses.id = grades.courseID, SPL_exams_questions.moed = grades.moed
(15) WHERE courses.name = 'SPL' and grades.moed = ?

ג.
(4) SELECT moed, numOfCorrectAnswers
(13) FROM grades
(15) WHERE grades.moed = ?

ד.
(4) SELECT numOfCorrectAnswers
(13) FROM grades JOIN courses
(15) WHERE courses.name = 'SPL' and grades.moed = ?

פתרון:

1 - א
2 - ב
3 - ג
4 - ד
5 - א
6 - ה
7 - ג
8 - ב
9 - ה
10 - ג
11 - ג
12 - ב
13 - ז
14 - ו
15 - א
16 - א
17 א