תשובות מספר נבחן:_____

על תשובות ריקות יינתן 20% מהניקוד!

שאלה 1 (30 נקודות)

סעיף א (4 נקודות)

PRE: none

POST: size() == @PRE(size() && Vi : itemAt(i) == @PRE(itemAt(i)) &&
@return is the sum of the items at @PRE

סעיף ב (6 נקודות)

בזמן שת'רד אחד מבצע sum, ת'רד שני מוריד איבר אחד ו/או מוסיף איבר אחר לתור (המתודה sum אינה מסונכרנת אז זה אפשרי). הסכום שיתקבל אינו סכום האברים שהיו ב PRE, בניגוד לתנאי הסיום.

סעיף ג (10 נקודות)

```
class ExtIntQueue extends SynchQueue<Integer> {

        private volatile int _version;

        ExtIntQueue() { _version = 0; }

        public synchronized void add(T item) {
                super.add(item);
                _version++;
        }

        public synchronized remove () {
                super.remove();
                _version++;
        }

        public int sum() {
                int s=0;
                int origVersion = version;
                for (int i=0; i<size(); i++) {
                   synchronized(this) {
                        if (version == origVersion)
                           s += get(i);
                        else
```

```
                    throw new ConcurrentModificationException();
                }
            }
            return s;
        }
}
```

סעיף ד (10 נקודות)

```
class Mystic implements Runnable  {
  ...
   protected void think() {

            Semaphore firstFork = (_id % 2 == 0 ? _leftFork : _rightFork);
            Semaphore secondFork = (_id % 2 == 0 ? _rightFork : _leftFork);

             // 1. acquire the forks
             firstFork.acquire(true);
             secondFork.acquire(true);

             // 2. Think (the 'thinking' is simulated by a busy wait)
            for (int i=0; i< 100000; i++);

            // 3. release the forks
            secondFork.release(true);
            firstFork.release(true);

   }

   protected void eat() throws InterruptedException {
             Semaphore firstFork = (_id % 2 == 0 ? _leftFork : _rightFork);
             Semaphore secondFork = (_id % 2 == 0 ? _rightFork : _leftFork);

             // 1. acquire the forks
             firstFork.acquire(false);
             secondFork.acquire(false);

            // 2. eat (the 'eating' is simulated by a busy wait)
            for (int i=0; i< 100000; i++);

            // 3. release the forks
            secondFork.release(false);
            firstFork.release(false);
   }
}



class Semaphore {
   private int _readers;
   private int _writers;
```

```
    public Semaphore() {
        _readers = 0;
        _writers = 0;
    }

    public synchronized void acquire(boolean bReader) throws InterruptedException {
        while (_writers > 0 || (!bReader && _readers >0))
            wait();
        (bReader ? _readers++ : _writers++);
    }

    public synchronized void release(boolean bReader) throws InterruptedException {
        (bReader ? _readers-- : _writers--);
        notifyAll();
    }
}
```

שאלה 2 (30 נקודות)

סעיף א (10 נקודות)

```
template <class T>
ITEM_COLLECTION<T>::ITEM_COLLECTION(const ITEM_COLLECTION &s)
{
	_size = s._size;
//הקצאה לא תיקנית בין -4 ל-5
	_items = new T[_size];
	for (int i = 0; i < _size; i++)
		*_items[i] = *s._items[i];
}
```

סעיף ב (5 נקודות)

```
void main()
{
//הקצאה של SHAPE לא תיקנית -3
	ITEM_COLLECTION<SHAPE*> myShapeCollection(50);
	for (int i = 0; i < 50; i++)
		myShapeCollection.Add(new SPHERE(), i);
}
```

סעיף ג (5 נקודות)

העתקה BYVAL משכפלת מצביעים בתוך _ITEMS. לכם ביציאה מהפונקציה יקרא הבנאי ההורס וימחק הזכרון המוצבע מMAIN. מי שדיבר על העתקת מצביעים אך לא התייחס לקריאה לבנאי הורס ביציאה מהפונקציה BYVAL בין -2 ל -5.

סעיף ד (10 נקודות)

```
void main()
{
	ITEM_COLLECTION<std::shared_ptr<SPHERE>> myShapeCollection(50);
```

```
for (int i = 0; i < 50; i++){
        std::shared_ptr<SPHERE> shPtr(new SPHERE());
        myShapeCollection.Add(shPtr, i);
}}
```

מי שהסביר עקרון SMART_PTR אך לא הראה שימוש בדוגמא בין -3 ל-6 נקודות

שאלה 3 (30 נקודות)

סעיף א (10 נקודות)

ProtocolTask מוכנסים לתור ב-executor ע"י ת'רד אחד בלבד, ה-Reactor, בפונקצית ה read של ה ConnectionHandler. סדר ההכנסה ל executor (וסדר הקריאה ל read) נבחר ע"י ה iterator בשורה:

Iterator<SelectionKey> it = selector.selectedKeys().iterator();

סדר זה אינו תלוי במספר הבקשות או במספר הבייטים, אלא ב key של ה SelectionKey של ה-Selector. בנוסף, ה-Reactor אינו לוקח בחשבון בשום שלב את העבר.

ניקוד מלא ניתן למי שנתן תרחיש נכון ונתן את אחד ההסברים האלה (או שההסבר עולה בבירור מתיאור התרחיש).

ישנן דוגמאות רבות לשני התרחישים. למשל:

תרחיש ב:

נניח שיש 2 לקוחות ו(בה"כ)לקוח א שלח יותר בקשות מלקוח ב. כעט נניח שה-Reactor מתעורר מפונקציית ה-select, כאשר שני הלקוחות שלחו מידע וה-iterator מכיל את ה-SelectionKey של שניהם: א ואח"כ ב.

ה-Reactor יקרא קודם ל read של ה-ConnectionHandler של לקוח א ואח"כ ל-read של ה-ConnectionHandler של לקוח ב.

תרחיש א:

אותו דבר כמו תרחיש ב למעט המשפט הראשון שעכשיו יהיה: "נניח שיש 2 לקוחות ושמספר הבייטים שנתקבלו ונשלחו ללקוח א גדולים יותר מלקוח ב".

סעיף ב (20 נקודות)

השינויים (לשני התרחישים יחד):

Executor: In the constructor we will change the first line to be:

```
        taskQueue = new PriorityBlockingQueue<Runnable>();
```

ProtocolTask:

Change class declaration:

```
        public class ProtocolTask<T> implements Runnable,
                                        Comparable<ProtocolTask<T>>
```

Add members:

```
        private volatile int _handledMessages = 0;
        private int _ioBytes = 0;
```

Add the method - compareTo:

```
        @Override
```

```
public int compareTo(ProtocolTask<T> o)}
        // First scenario:
                // return _ioBytes - o._ioBytes;
        // Second scenario:
                // return _handledMessages - o._handledMessages;
}
```

Add the method - addToIoBytes:

```
public void addToIoBytes(int i) {
        _ioBytes += i;
}
```

Method addBytes - add the following line:

```
addToIoBytes(b.remaining());
```

```
Method run:
        Add the following line just before the while ends (after the
        catch):
                _handledMessages++;


ConnectionHandler:
    Method write –
        replace
                _sChannel.write(buf);
        with:
                _task.addToIoBytes(_sChannel.write(buf));
```

שימו לב:
במקרה של תרחיש א אין צורך בשום סנכרון כיוון שהת'רד היחיד שניגש לשדה ioBytes_הוא הת'רד של ה-Reactor. (השינוי שלו מתבצע במהלך הפונ' - read/write והקריאה שלו מתבצעת במהלך ההכנסה לתור של ה executor בפונ' compareTo, גם בפונ' ה read).

במקרה של תרחיש ב עדכון השדה handledMessages_ מתבצע ע"י הת'רד של ה-ProtocolTask, ואילו הקריאה שלו מתבצעת ע"י הת'רד של ה-reactor במהלך הכנסת ה-ProtocolTask לרשימה. לכן יש אפשרות בתרחיש זה להשתמש ב volatile/AtomicInteger על השדה או לחלופין לסנכרן את compareTo ואת שינוי השדה על מנעול השונה מ this (נעילה על this תגרור עצירה של הת'רד של ה-Reactor לזמנים ארוכים).

ניקוד חלקי ניתן על סנכרון מיותר, לא יעיל או לא נכון.

כמו-כן, שימו לב שביקשנו בא' כמה בייטים **נקראו ונכתבו**. לכן הוספה של buf.size ב write אינה נכונה כי לא בהכרח נשלח את הבייטים האלה. בדומה הוספה של גודל ההודעה ב run של ה-ProtocolTask לא מספקת כי יכולים להיות עוד בייטים שקראנו ולא מכילים הודעה שלמה או עוד הודעות שה-ProtocolTask לא עיבד אבל כבר נקראו.

אין צורך לשמור את כמות ההודעות הכוללת כיוון ש: $\forall Y > 0: \frac{X_1}{Y} \leq \frac{X_2}{Y} \Leftrightarrow X_1 \leq X_2$. לא ירד ניקוד על שמירת מספר ההודעות הכללי (אם נעשה נכון).

שאלה 4 (10 נקודות)

סעיף א (2 נקודות)

```
INSERT INTO Participants (ID, Name, Role) VALUES ('111111111', 'Kim Philby', 'double agent');
```

סעיף ב (4 נקודות)

אחת הדרכים להרחבת ה-data model הנתון המאשפרת הגבלת תפקידי המשתתפים בהקלטות (כפי שמתואר בשאלה) היא:

```
CREATE TABLE Roles (
   Id INTEGER PRIMARY KEY,
   Role VARCHAR(30) );

ALTER TABLE Participants
          DROP COLUMN Role;

ALTER TABLE Participants
          ADD RoleId INTEGER;

ALTER TABLE Participants
          ADD FOREIGN KEY (RoleId)
          REFERENCES Roles(Id);

INSERT INTO Roles (Id, Role) VALUES (1, 'vocals');
INSERT INTO Roles (Id, Role) VALUES (2, 'guitar');
INSERT INTO Roles (Id, Role) VALUES (3, 'bass');
INSERT INTO Roles (Id, Role) VALUES (4, 'keyboard');
INSERT INTO Roles (Id, Role) VALUES (5, 'drums');
INSERT INTO Roles (Id, Role) VALUES (6, 'technician');
INSERT INTO Roles (Id, Role) VALUES (7, 'producer');
```

סעיף ג (4 נקודות)

```
SELECT Song
     FROM Tapes JOIN TapeParticipants ON Tapes.Id = TapeId
            JOIN Participants ON ParticipantId = Participants.Id
            JOIN Roles ON RoleId = Roles.Id
     WHERE Band = 'The Beatles' AND
         Name = 'Phil Spector' AND Role = 'producer'
     ORDER BY Take;
```