תשובות מספר נבחן:_____

על תשובות ריקות יינתן 20% מהניקוד!

שאלה 1 (30 נקודות)

בשאלה זו נדרשתם להשתמש בכל הכללים של DBC למעט הכלל הראשון של הפרדת שאילתות מפקודות.
לא הורדנו נקודות על הגדרה לא נכונה של isEmpty ו peek. כמו-כן לא ירדו נקודות על שימוש ב isEmpty ו-isEmpty! במקום size()==0 ו size>0 ב pre/post/inv. ניקוד חלקי ניתן גם למי שהגדיר את lastPeek כפונ' עזר במקום itemAt.

```
/** @inv: size()>=0 **/
interface Queue<T> {
        /** Returns the number of elements in the queue. **/
        int size();

        /** Retrieves, but does not remove, the i-th element of this queue.
 * Throws NullPointerException if i >= size().
 *
 * @pre: 0 <= i < size() **/
        T itemAt(int i);

        /** Adds the given item to the queue. Throws NullPointerException if
         * the given item is null.
         *
         * @pre: item != null
         * @post: size()=@pre(size())+1 && itemAt(size()-1)=item  **/
        void add(T item);

        /** Removes the head item of this queue.
 * Throws NoSuchElementException if the queue is empty.
         *
         * @pre: size()>0
         * @post: size()=@pre(size())-1
         * @post: if(@pre(size())>1) {itemAt(0)=@pre(itemAt(1))}
 * The last post condition may be dropped if itemAt is defined,
 * see the lecture notes. **/
        void remove();

/** Retrieves, but does not remove, the head of this queue, or
 * returns null if this queue is empty.
         *
         * @post: @return(size()==0 ? null : itemAt(0)          **/
```

```
        T peek();

        /** Returns true iff this queue contains no elements.
         *
         * @post: @return(size()==0)          **/
        boolean isEmpty();
}
```

סעיף ב (6 נקודות)

בסעיף זה נדרשתם לבדוק את נכונות הפונקציות (מקרי הקצה, ו-post-conditions), תוך השענות על נכונות הפונקציות האחרות. נדרשו לפחות שתי הכנסות והוצאות בשביל לבדוק את נכונות התור.

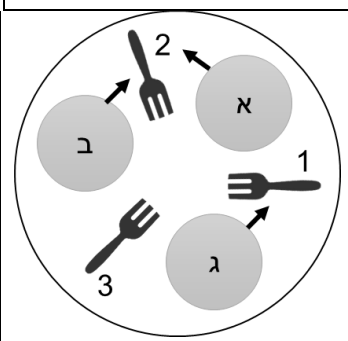
```
public class QueueTest {
	private static final int TEST_SIZE = 20;

	...

	@Test
	public void testAdd() {
		try {
			queue.add(null);
			fail("add(null) did not throw NullPointerException.");
		} catch (NullPointerException e) {
			// The test passed, nothing to check here.
		}

		// It is enough to check for two items only.
		for (int i = 0; i < TEST_SIZE; i++) {
			String item = Integer.toString(i);
			queue.add(item);
			assertEquals(i + 1, queue.size());
			assertEquals(item, queue.itemAt(i));
		}
	}

	@Test
	public void testRemove() {
		try {
			queue.remove();
			fail("remove() with empty stack did not throw NoSuchElementException.");
		} catch (NoSuchElementException e) {
			// The test passed, nothing to check here.
		}

		// It is enough to check for two items only.
		for (int i = 0; i < TEST_SIZE; i++)
			queue.add(Integer.toString(i));
		for (int i = 0; i < TEST_SIZE -1; i++) {
			String t = queue.itemAt(1);
			queue.remove();
```

```
                    assertEquals(TEST_SIZE - i - 1, queue.size());
                    assertEquals(t, queue.itemAt(0));
                    // The last assert may be dropped. See the lecture notes.
            }
            queue.remove();
            assertEquals(0, queue.size());
    }
}
```

סעיף ג (6 נקודות)

No - Deadlock No - Livelock

Starvation – בסעיף זה אתם מתבקשים לתת דוגמא ולא לומר אמירה כללית.
הבעיה היא כמובן שסדר תפיסת המנעולים אינו הוגן. אמירות כלליות (זמן CPU לא שווה ועוד) ו/או סדר פעולות שלא מצינות את הבעיה קיבלו במקרה הטוב ניקוד חלקי.



הסתכלו על הציור הבא ועל סדר הפעולות הבא:

1. אחרי שהוא חושב, פילוסוף ב' לוקח את מזלג 2 ואז את מזלג 3 ומתחיל לאכול (פילוסוף א' שגם אמור להתחיל עם מזלג 2 לא יכול להתחיל ומבצע wait על מזלג 2).
2. פילוסוף ג' לוקח את מזלג 1.
3. פילוסוף ב' מסיים לאכול ומשחרר את 3 ו-2 ומעיר את פילוסוף א' שממתין.
4. למרות שהעירו אותו, ה scheduler לא נותן לפילוסוף א' זמן לרוץ אלא דווקא לפילוסוף ג' שלוקח את מזלג 3 ומתחיל לאכול. (לא פה הבעיה).
5. פילוסוף ג' מסיים לאכול ומשחרר את 3 ו-1.
6. ה scheduler נותן לפילוסוף ב' זמן ריצה וזה מסיים לחשוב ולוקח שוב את 2 ו-3 (סדר תפיסת המנעולים אינו הוגן ולמרות שב' הגיע ל-2 אחרי א' – הוא לוקח את המפתח).
7. פילוסוף ג' מסיים לחשוב ושוב לוקח את 1.
8. חוזר חלילה.

כך פילוסוף א' נשאר רעב.

סעיף ד (10 נקודות)

בסעיף זה העיקר היה להבין שההכנסה לתור נעשית ללא סנכרון כדי ליצור הוגנות.
היו שלא הבינו מה המשמעות של להיות הוגנים. הכוונה לא לאכוף round robin כי אנחנו נהפוך לריצה סדרתית – ת'רדים יכולים לקחת מזלג ולנצל את הזמן לאכילה אבל הם נאלצים לחכות שמישהו שתורו יסיים לחשוב.
הוגנות הכוונה שמי שהגיע ראשון ל sync - נכנס.

```
class FairSemaphore {
        private final int _permits; private int _free;
        private final ConcurrentLinkedQueue<Thread> _threads;

        public FairSemaphore(int permits) {
                _permits = permits;
                _free = permits;
                _threads = new ConcurrentLinkedQueue<>();
```

```
	}
	public void acquire() throws InterruptedException {
		Thread current = Thread.currentThread();
		_threads.add(current);
		synchronized (this) {
			while (_threads.peek() != current || _free <= 0)
				wait();
			_free--;
			_threads.remove();
			notifyAll();
		}
	}
	public synchronized void release() throws InterruptedException {
		if (_free < _permits) {
			_free++;
			notifyAll();
		}	}
}
```

שאלה 2 (30 נקודות)

הקוד של סעיף א + סעיף ב:

```
#include<stdio.h>
#include<windows.h>

class SHADER;
class SPHERE;
class BOX;
class SHAPE
{
public:
        virtual void RenderMe(SHADER *s) = 0;
        virtual SHAPE* Clone()  = 0;
};

class SHADER
{
public:
        virtual void Render(SHAPE *s) = 0;
        virtual void Render (SPHERE *s) = 0;
        virtual void Render (BOX *b) = 0;
        virtual SHADER* Clone()  = 0;
};

class SPHERE: public SHAPE
{
public:
        void RenderMe(SHADER *s) {s->Render(this);}
        SPHERE(const SPHERE &t){;}
        SHAPE* Clone() {return new SPHERE(*this);}
};

class BOX: public SHAPE
{
public:
        BOX() { ; }
        void RenderMe(SHADER *s) {s->Render(this);}
        BOX(const BOX &t){;}
        SHAPE* Clone() {return new BOX(*this);}
private:
};
```

```
class BASIC_SHADER : public SHADER
{
public:
	BASIC_SHADER() { ; }
	BASIC_SHADER(const BASIC_SHADER &t){;}
	SHADER* Clone() {return new BASIC_SHADER(*this);}
	void Render(SHAPE *s) {s->RenderMe(this);}
	void Render (SPHERE *s) {;}
	void Render (BOX *b) {;}
private:
};
class COMPLEX_SHADER : public SHADER
{
public:
	void Render(SHAPE *s) {s->RenderMe(this);}
	COMPLEX_SHADER(const COMPLEX_SHADER &t){;}
	SHADER* Clone() {return new COMPLEX_SHADER(*this);}
	void Render (SPHERE *s) {;}
	void Render (BOX *b) {;}
};

class RENDER_MGR
{
public:
	RENDER_MGR(int n);
	RENDER_MGR(const RENDER_MGR &m);
	int Render();

private:
	SHADER **_r;
	SHAPE **_s;
	int		_N;
};


int RENDER_MGR::Render()
{
	for (int i=0; i<_N; i++)
		_r[i]->Render(_s[i]);
	return 0;
}

RENDER_MGR::RENDER_MGR(const RENDER_MGR &m)
{
```

```
	_N = m._N;
	_r = new SHADER*[_N];
	_s = new SHAPE*[_N];
	for (int i=0; i<_N; i++)
	{
		_r[i] = m._r[i]->Clone();
		_s[i] = m._s[i]->Clone();
	}
}

RENDER_MGR::RENDER_MGR(int n)
{
	_N = n;
	_r = new SHADER*[_N];
	_s = new SHAPE*[_N];
	for (int i=0; i<n; i++)
	{
		_r[i] = new BASIC_SHADER();
		_s[i] = new BOX();
	}
}

void main()
{
	RENDER_MGR mgr(2);
	mgr.Render();
}
```

מפתח ניקוד:

א:'
טעות פוינטר ,THIS, חץ\נקודה2- &\
טעות אי שימוש ב void RenderMe(SHADER *s) ובמקום שימוש ב void RenderMe(COMPLEX_SHADER *s)
הורדו -4

ב'
טעות אי העתקה עמוקה -10
טעות new SHAPE או new SHADER הורדו -10

שאלה 3 (30 נקודות)

סעיף א (6 נקודות)

```
class MultipleClientProtocolServer implements Runnable {
  …
  public void run()    {
    …
    ExecutorService executor = Executors.newFixedThreadPool(9);
    while (true) {
        try {
          ConnectionHandler newConnection = new ConnectionHandler(serverSocket.accept(), factory.create());
          new Thread(newConnection).start();
          executor.execute(newConnection);
        } catch (IOException e) {
           …
        }
      }
   }
   …
}
```

סעיף ב (6 נקודות)

```
public class Printer_Stub implements Printer {
  …
  public void print(String str) throws RemoteException {
    try {
      …
      String msg = str + "\n";
      String msg = str;
      socket.getOutputStream().write(encodeMessageLength(msg));
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      msg = "bye\n";
      msg = "bye";
      socket.getOutputStream().write(encodeMessageLength(msg));
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      …
    } catch (Exception e) {
      …
    }
  }
  private static byte[] encodeMessageLength(String msg) { … } // returns a byte array holding the UTF-8
encoding of msg.getBytes("UTF-8").length
}
class ConnectionHandler implements Runnable {

   public void process() throws IOException {
      String msg;

      while ((msg = readMessage()) != null)   {
         …
      }
   }
  private String readMessage() {
    char[] message = new char[decodeMessageLength()];
    in.read(message);
    return new String(message);
  }
  private int decodeMessageLength() { … } // reads from "this.in" 4 bytes representing an integer
}
```

סעיף ג (6 נקודות)

```
public class Printer_Stub implements Printer {
  …
  public void print(String str) throws RemoteException {
    try {
      Socket socket = new Socket(_skelHost,_skelPort);
      InetAddress address = InetAddress.getByName(_skelHost);
      DatagrapSocket socket = new DatagramSocket();
      String msg = str + "\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      byte[] buf = msg.getBytes("UTF-8");
      DatagramPacket packet = new DatagramPacket(buf, buf.length, address, _skelPort);
      socket.send(packet);

      msg = "bye\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      socket.close();
    } catch (Exception e) {
      throw new RemoteException(e.toString());
    }
  }
}
```

```
interface ServerProtocol {
  String processMessage(String msg);
  boolean isEnd(String msg);
}

interface ServerProtocolFactory {
  ServerProtocol create();
}

class ConnectionHandler implements Runnable {…}
class MultipleClientProtocolServer implements Runnable {
   private ServerSocket serverSocket DatagramSocket serverSocket;
   private int listenPort;
   private ServerProtocolFactory factory;

   public MultipleClientProtocolServer(int port, ServerProtocolFactory p)   {
      serverSocket = null;
      listenPort = port;
      factory = p;
   }
```

```
  public void run()    {
    try {
      serverSocket = new ServerSocket(listenPort);
      serverSocket = new DatagramSocket(listenPort);
      System.out.println("Listening...");
    }
    catch (IOException e) {
      System.out.println("Cannot listen on port " + listenPort);
    }
    byte[] buf = new byte[Integer.MAX_VALUE];
    ServerProtocol protocol = factory.create();
    while (true) {
      try {
        ConnectionHandler newConnection = new ConnectionHandler(serverSocket.accept(), factory.create());
        new Thread(newConnection).start();
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);
        String response = protocol.processMessage(msg);
        if (response != null) {
          out.println(response);
          InetAddress address = packet.getAddress();
          int port = packet.getPort();
          byte[] buf = response.getBytes("UTF-8");
          packet = new DatagramPacket(buf, buf.length, address, port);
          socket.send(packet);
        }
      } catch (IOException e) {
         System.out.println("Failed on port " + listenPort);
      }
    }
  }
  …
}
```

סעיף ד (6 נקודות)

```
public interface Math extends java.rmi.Remote {
   // Receives a list of numbers and returns a list of their root numbers.
   List<Float> sqrt(List<Integer> numbers) throws java.rmi.RemoteException;
}
// here is a possible implementation of the actual server – although it was not expected as part of your answer…
public class MathImpl  extends java.rmi.server.UnicastRemoteObject  implements Math  {
  public MathImpl() throws java.rmi.RemoteException { }
  List<Float> sqrt(List<Integer> numbers) throws java.rmi.RemoteException {
     List<Float> result = new ArrayList<Float>(numbers.size());
     for (int n : numbers) {
       result.add(new Float(java.lang.Math.sqrt(n)));
     }
     return result;
  }
}
// and here are the key elements in a possible implementation of the expected skel + stub…
class MathProtocol implements ServerProtocol {
   Math _math;
   public MathProtocol (Math math) { _math = math; }
   public String processMessage(String msg) {
       List<Integer> numbers = decodeListOfIntegers(msg);
       List<Float> squareRoots = _math.sqrt(numbers);
        return encodeAsString(squareRoots);
   }
   private static List<Integer> decodeListOfIntegers(String string) { … }
   private static String encodeAsString(List<Float> numbers) { … }
}
```

```
public class Math_Skel {
  Math _Skel(Math math) throws Exception {
     ServerProtocolFactory  protocolFactory = new ServerProtocolFactory() {
       public ServerProtocol create() {
          return new MathProtocol(math);
       }
      };
      MultipleClientProtocolServer server = new MultipleClientProtocolServer(1984, protocolFactory);
      Thread serverThread = new Thread(server);
       serverThread.start();
  }
}
```

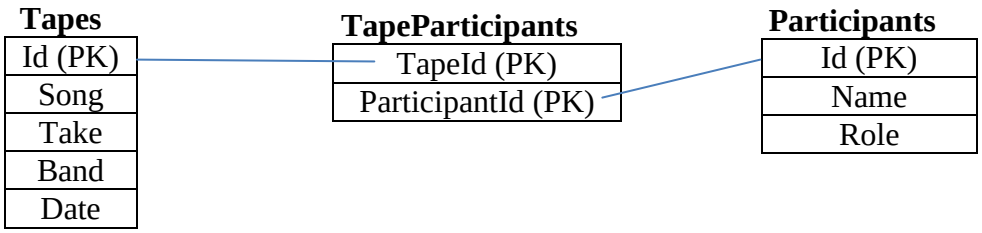
```
public class Math_Stub implements Math {
  String  _skelHost;
  int _skelPort;

  Math_Stub(String skelHost,int skelPort) throws RemoteException {
    _skelHost = skelHost; _skelPort = skelPort;
  }

  List<Float> sqrt(List<Integer> numbers) throws java.rmi.RemoteException {
    try {
      Socket socket = new Socket(_skelHost,_skelPort);
      String msg = encodeAsString(numbers) + "\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      List<Float> result = DecodeListOfFloats(new BufferedReader(
          new InputStreamReader(socket.getInputStream(),"UTF-8")).readLine());
      msg = "bye\n";
      socket.getOutputStream().write(msg.getBytes("UTF-8"));
      socket.close();
      return result;
    } catch (Exception e) {
      throw new RemoteException(e.toString());
    }
  }
  private static List<Float> decodeListOfFloats(String string) { … }
  private static String encodeAsString(List<Integer> numbers) { … }
}
```

סעיף ה (6 נקודות)

```
public class Printer_Skel {
  private final static int poolSize = 10;
  Printer_Skel(Printer printer) throws Exception {
    ~~ServerProtocolFactory protocolFactory = new ServerProtocolFactory() {~~
      ~~public ServerProtocol create() {~~
        ~~return new PrinterProtocol(printer);~~
      ~~}~~
    ~~};~~
    ~~MultipleClientProtocolServer server = new MultipleClientProtocolServer(1984, protocolFactory);~~
    static class AsyncPrinterProtocol implements AsyncServerProtocol<StringMessage> {
      Printer _printer;
      private boolean terminated = false, shouldClose = false;
      AsyncPrinterProtocol(Printer printer) { this.printer = printer; }
      boolean shouldClose() { return shouldClose; }
      public void connectionTerminated() { terminated = true; }
      public StringMessage processMessage(StringMessage msg) {
        if (connectionTerminated) return null;
        if (isEnd(msg)) shouldClose = true;
        printer.print(msg);
        return new StringMessage("Your message has been printed.");
      }
      public boolean isEnd(StringMessage msg) { return msg.equals("bye"); }
    }
    Runnable server = new Reactor<StringMessage>(1984, poolSize,
      () -> new AsyncPrinterProtocol(printer),
      () -> new FixedSeparatorMessageTokenizer("\n", Charset.forName("UTF-8")));
    Thread serverThread = new Thread(server);
    serverThread.start();
  }
}
```

שאלה 4 (10 נקודות)

סעיף א (5 נקודות)



סעיף ב (5 נקודות)

```
SELECT Tapes.Song, Tape.Take

FROM   Tapes INNER JOIN TapeParticipants ON Tapes.Id = TapeParticipants.TapedID
       INNER JOIN Participants ON Participants.Id = TapeParticipants.Id

WHERE Tapes.Date >= '1/1/1969' AND  Tapes.Date <= '31/1/1969' AND
       Tapes.Band = 'The Beatles' AND
       Participants.Name = 'Billy Preston' AND
       Participants.Role = 'Keyboards'

ORDER BY  Tapes.Date ASC
```