גיליון תשובות מספר נבחן:_____

על תשובות ריקות יינתן 20% מהניקוד!

שאלה 1 (30 נקודות)

סרגל הבדיקה:
הורדו נקודות רק על פרטים שחסרים או שנרשמו בניגוד לפתרון הנתון בהמשך. לא ניתנו ולא ירדו נקודות על כל מיני תוספות או דברים מיותרים.

סעיף א: שתי נקודות על בדיקת השקה בין גלגלים שכנים. שתי נקודות על בדיקת מיקום גלגל שכן בצד הנכון - שמאלי או ימני. נקודה אחת על רדיוס חיובי.

סעיף ב: נקודה אחת על כל מטודה. הנקודה ירדה רק אם המטודה מומשה שלא בהתאם למיקום בקוד בו היא מופעלת.

סעיף ג: שתי נקודות על התשובה "בטוח" ושתי נקודות על הנימוק. ירדו כל ארבע הנקודות על תשובה "לא בטוח", בפרט אם בתרחיש שניתן כנימוק מתואר מצב "לא נכון" במקום "לא בטוח".

סעיף ד: שתי נקודות על התוספת לאינוורינטה - ירדו אם אין התייחסות לטווח הערכים של _state. נקודה אחת על התשובה "בטוח". שתי נקודות על הנימוק. לא הורדו נקודות על תרחיש כנימוק לתשובה "לא בטוח" אם הוא זהה לתרחיש של סעיף ג.

סעיף ה: נקודה אחת על תנאי התחלה. שלש נקודות על תנאי סיום, ירדו שלש נקודות אם אין בדיקה כמותית של ערכו החדש של _state. נקודה אחת על השימוש ב- offset. נקודה אחת על השימוש בכוון הסיבוב clockwise.

סעיף ו: שתי נקודות על התשובה "לא נכון". שלש נקודות על תרחיש המנמק את התשובה, מתוכן ירדו נקודה או שתיים אם התרחיש אינו מפורט מספיק, או שאינו מתאים לסימולציה הספציפית שבשאלה. חמש נקודות על הסינכרון, מתוכן שלש על פירוט של אמצעי הסנכרון ומיקומו בקוד.

סעיף א (5 נקודות)

```
@inv _radius > 0
@inv _left==null || _left.getX()<_x  //correct position on left side
@inv _right==null || _right.getX()>x  //correct position on right side
@inv _left==null || verifyEuclidianDistance(_left) //gears must touch
@inv _right==null || verifyEuclidianDistance(_right)
```

הפונקציה verifyEuclidianDistance מחזירה ערך בוליאני. ראה מימוש בסעיף ב.

סעיף ב (2 נקודות)

```
protected boolean check(float x, float y, float radius) {
        return (radius>0);
        //x and y can be any values – no need to check anything
}

protected boolean check(Gear other, boolean whichSide) {
        if (other==null) return false;
        if (whichSide && (other,getX()>=x)) return false;
        if (!whichSide && (other,getX()<=x)) return false;
        return verifyEuclidianDistance(other);
}

protected boolean verifyEuclidianDistance(Gear other) {
        if (other==null) return false;
        float sideX=_x-other.getX();
        float sideY=_y-other.getY();
        float distance=_radius+other.getRadius();
        return sidex*sideX+sideY*sideY==distance*distance;
}
```

סעיף ג (4 נקודות)

המחלקה בטוחה כי האינוורינטה תמיד נשמרת.
רק הבנאי והמטודות sertRight(), setLeft() משנים שדות רלוונטיים לאינוורינטה. כל שינוי נעשה אחרי הפעלת אחת הבדיקות מסעיף ב ונזרקת חריגה במקרה של הפרה.

סעיף ד (5 נקודות)

התוספת לאינוורינטה: 0<=_state<360 (אפשר להניח שבממשק Gear קיימת מטודה getState).
המחלקה עדיין בטוחה. כל שינוי בערכו של _state מתבצע תחת modulu של 360.

סעיף ה (4 נקודות)

```
@pre none
@post (clockwise && (_state==(@pre(_state)+getOffset(_radius))%360)) ||
        (!clockwise && (_state==(@pre(_state)-getOffset(_radius))%360))
```

סעיף ו (10 נקודות)

ההרצה הנתונה אינה נכונה.
להלן תרחיש המדגים את הבעיה.
נקרא לחוטים שבסימולציה T1 T2. בשניהם gear_ מתייחס לאותו גלגל.
שני החוטים במצב שלפני ההפעלה ראשונה של המטודה ()gear.move_.
נניח בלי הגבלת כלליות שהקריאה getOffset(_radius) מחזירה 1.

T1 מפעיל את gear.move(true)_, קורא את ערכו הקודם 0 של state_. מתבצע חילוף קונטקסט
T2 מפעיל את gear.move(false)_ ומבצע את כל המטודה עד סופה. state_ מקבל ערך חדש 359. מתבצע חילוף קונטקסט.
T1 ממשיך בתוך move ומבצע את שאר המטודה לפי הערך הקודם 0 של state_ אותו הספיק לקרוא לפני החילוף. state_ מקבל ערך חדש 1.

התוצאה היא שאחרי שתי הפעלות של סיבוב הגלגל בכוונים הפוכים החל ממצב התחלתי 0, מצבו של הגלגל הוא 1 ולא 0. כלומר, "הלך לאיבוד" אחד הסיבובים. זה לא יכול לקרות בהפעלה סדרתית.

הבעיה היא חוסר תאום בין החוטים לגבי סיבוב הגלגל. נפתור זאת על ידי סינכרון המטודה move.

להלן שתי אפשרויות תיקון עבור הסימולציה הנתונה (יש עוד, כל אפשרות רלוונטית התקבלה).
1. הוספת המילה synchronized לחתימת המטודה move בממשק Gear ובמחלקה SimpleGear.
2. סינכרון בלוק על האובייקט gear_ בתוך המטודה ()run במחלקה Spinner.

```
synchronized(_gear) {
    _gear.move(_clockwise);
}
```

לפיכך חוט שנכנס למטודה move יסיים אותה באופן מלא (כולל כל הקריאות הפנימיות הרקורסיביות של moveLeft ושל moveRight), ורק אז חוט אחר יכול להיכנס.

הערה: הפתרון לא בהכרח עובד נכון בסימולציה אחרת מזו הנתונה בשאלה.

שאלה 2	**(30 נקודות)**

סעיף א (15 נקודות)

```
template <class T> class node{
public:
	node():_data(){;}
	node(const T &val):_data(val){;}
	~node(){;}
	const T& getData(){return _data;}
	node* getNext(){return _next;}
	void  setNext(node*n){_next=n;}
private:
	const T&  _data;
	node *_next;
};
template <class T> class mvector{
public:
	mvector();
	~mvector();
	mvector(const mvector &);
	mvector & operator=(const mvector &);
	unsigned getSize();
	void pushAt(const T &, unsigned );
	const T & popAt(unsigned);
	void copy(const mvector &);
private:
	unsigned _len;
	node<T> *_head;
};
template <class T>
mvector<T>::mvector(){
	_head=0;
}
template <class T>
mvector<T>::~mvector(){
	node<T> *_curr;
	node<T> *_prev = _head;

	for (unsigned i=0; i<_len; i++){
		_curr = _prev->getNext();
		delete _prev;
		_prev = _curr;
	}
}
template <class T>
void mvector<T>::copy(const mvector &other){
	_len = other._len;
	node<T> *_prev, *_curr;
	node<T> *_curr_other = other._head;

	for (unsigned i=0; i<_len; i++)   {
		_curr = new node<T>(_curr_other->getData());

		if (i)
			_prev->setNext(_curr);
		else
```

```
                        _head = _curr;

                _curr_other = _curr_other->getNext();
                _prev = _curr;
        }
}
template <class T>
mvector<T>::mvector(const mvector & v){
        this->copy(v);
}
template <class T>
mvector<T> & mvector<T>::operator=(const mvector &other){
        this->copy(other);
        return *this;
}
template <class T>
unsigned mvector<T>::getSize(){
        return _len;
}
template <class T>
void mvector<T>::pushAt(const T& val, unsigned pos){
        node<T> *_prev = _head;
        node<T> *_curr;

        for (unsigned i=1; i<pos; i++){
                _curr = _prev->getNext();
                if (_curr==0){
                        _curr = new node<T>();
                        _prev->setNext(_curr);
                }
                _prev = _curr;
        }

        _curr = new node<T>(val);

        if (pos==0)
                _curr->setNext(_prev);
        else
                _prev->setNext(_curr);
}
template <class T>
const T & mvector<T>::popAt(unsigned pos){
        node<T> *_curr = _head;
        node<T> *_tmp;

        for (unsigned i=0; i<pos; i++){
                _curr = _curr->getNext();
        }

        _tmp = _curr->getNext();
        _curr->setNext(_curr->getNext()->getNext());

        return _tmp->getData();
}
```

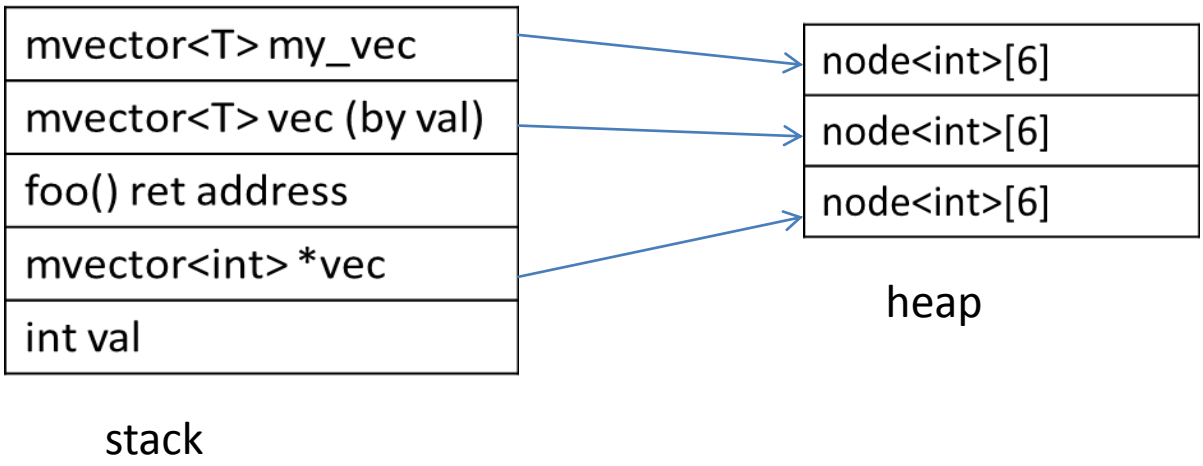
מימוש בנאים (deep copyאו shallow), אופרטור מעתיק, בנאי הורס = 5 נק.
מימוש insertAt 5נק.
מימוש popAt 5נק.

סעיף ב (5 נקודות)
3

סעיף ג (10 נקודות)

הורדו 2 נקודות על כל אחת מהשגיאות:
שימוש בVTABLE
חסר ret address
פרמטר לא הועבר BY VAL
אי הקצאה דינאמית של הוקטור על הHEAP

התקבלו מימושים עבור shallow copy



שאלה 3 (30 נקודות)

סעיף א (20 נקודות)

כדי לשנות את התקשורת עם הלקוחות מ TCP ל UDP נדרש:

- הסרת ה ServerSocketChannel
- הסרת ה SocketChannel
- הגדרת DatagramSocketChannel
- רישום ה DatagramSocketChannel ב Selector על אירועי READ או READ | WRITE
- הסרת ה ConnectionAcceptor
- הגדרת ConnectionHandler אחד עבור כל הלקוחות, כאשר מתודת ה read קוראת ב non-blocking מה DatagramSocketChannel כמה הודעות שאפשר, ומתודת ה write שולחת ב non-blocking דרך ה DatagramSocketChannel כמה הודעות שאפשר (ווקטור הפלט במקרה זה מכיל את כל ההודעות שצריכות להישלח ללקוחות השונים, עם כתובתם).
- העברת הכתובת של הלקוח (המתקבלת כערך חוזר של receive) ל ProtocolTask כך שניתן יהיה לשלוח לו את התשובה אחר כך (ע"י העברת התשובה עם כתובת הלקוח לווקטור הפלט ב ConnectionHandler). שימו לב כי UDP אינו מבטיח שמירה על סדר קבלת הודעות, כך שאין צורך לדאוג לכך ב ProtocolTask.
- ביטול ה Tokenizer (אין צורך בו, שהרי UDP מספק כבר את ההודעה כפי שנשלחה)

אי התייחסות לכל אחד מסעיפים אלו גררה הורדה של 2 נקודות (למעשה ירד בדרך כלל פחות, לפנים משורת הדין)

```
public class Reactor<T> implements Runnable {

    ...
    private final TokenizerFactory<T> _tokenizerFactory;
    private final EncoderFactory<T> _encoderFactory;

    public Reactor(int port, int poolSize, ServerProtocolFactory<T> protocol, TokenizerFactory<T> tokenizer) {
    public Reactor(int port, int poolSize, ServerProtocolFactory<T> protocol, EncoderFactory<T> encoder) {
        _port = port;        _poolSize = poolSize;
        _protocolFactory = protocol;
        _tokenizerFactory = tokenizer;
        _encoderFactory = enoder;
        }

private ServerSocketChannel createServerSocket(int port)
            ...
}

private DatagramSocketChannel createDatagramSocketChannel(int port)
```

```
        throws IOException {
    try {
        DatagramSocketChannel dgChannel = DatagramSocketChannel.open();
        dgChannel.configureBlocking(false);
        dgChannel.socket().bind(new InetSocketAddress(port));
        return dgChannel;
    } catch (IOException e) {
        logger.info("Port " + port + " is busy");
        throw e;
    }
}

public void run() {
    ExecutorService executor = Executors.newFixedThreadPool(_poolSize);
    Selector selector = null;
    ~~ServerSocketChannel ssChannel = null;~~
    DatagramSocketChannel dsChannel = null;

    try {
        selector = Selector.open();
        ~~ssChannel = createServerSocket(_port);~~
        dgChannel = createDatagramSocketChannel(_port);
    } catch (IOException e) {
        logger.info("cannot create the selector -- server socket is busy?");
        return;
    }

    ~~_data = new ReactorData<T>(executor, selector, _protocolFactory, _tokenizerFactory);~~
    _data = new ReactorData<T>(executor, selector, _protocolFactory, _encoderFactory);
    ~~ConnectionAcceptor<T> connectionAcceptor = new ConnectionAcceptor<T>( ssChannel, _data);~~
    ConnectionHadler<T> connectionHandler = new ConnectionHandler<T>(dgChannel, _data);

    try {
        ~~ssChannel.register(selector, SelectionKey.OP_ACCEPT, connectionAcceptor);~~
        dgChannel.register(selector, SelectionKey.OP_READ, connectionHandler);
    } catch (ClosedChannelException e) {
        logger.info("server channel seems to be closed!");
        return;
    }

    while (_shouldRun && selector.isOpen()) {
        try {
            selector.select();
        } catch (IOException e) {
            logger.info("trouble with selector: " + e.getMessage());
```

```
            continue;
        }

        Iterator<SelectionKey> it = selector.selectedKeys().iterator();
        while (it.hasNext()) {
            SelectionKey selKey = (SelectionKey) it.next();
            it.remove();

            if (selKey.isValid() && selKey.isAcceptable()) {
                logger.info("Accepting a connection");
                ConnectionAcceptor<T> acceptor = (ConnectionAcceptor<T>) selKey.attachment();
                try {
                    acceptor.accept();
                } catch (IOException e) {
                    logger.info("problem accepting a new connection: "+ e.getMessage());
                }
                continue;
            }
            if (selKey.isValid() && selKey.isReadable()) {
                ConnectionHandler<T> handler = (ConnectionHandler<T>) selKey.attachment();
                logger.info("Channel is ready for reading");
                handler.read();
            }
            if (selKey.isValid() && selKey.isWritable()) {
                ConnectionHandler<T> handler = (ConnectionHandler<T>) selKey.attachment();
                logger.info("Channel is ready for writing");
                handler.write();
            }
        }
    }
    stopReactor();
}

...

public static Reactor<StringMessage> startEchoServer(int port, int poolSize){
    ...
    final Charset charset = Charset.forName("UTF-8");
    EncoderFactory<StringMessage> encoderMaker = new EncoderFactory<StringMessage>() {
        public MessageEnocder<StringMessage> create() {
            return new EncoderMessageEncoder(charset);
        }
    };
    ...
Reactor<StringMessage> reactor =
```

```
            new Reactor<StringMessage>(port, poolSize, protocolMaker, tokenizerMaker ,encoderMaker);

    }
}
```

```
public class ConnectionHandler<T> {

    private static final int BUFFER_SIZE = 1024;
    protected final SocketChannel _sChannel;
    protected final DatagramSocketChannel _dgChannel;
     protected final ReactorData<T> _data;
    protected final AsyncServerProtocol<T> _protocol;
    protected final MessageTokenizer<T> _tokenizer;
    protected final MessageEncoder<T> _encoder;
    protected Vector<ByteBuffer> _outData = new Vector<ByteBuffer>();
    protected Vector<Pair< ByteBuffer,SocketAddress>> _outData;
     protected final SelectionKey _skey;
     private static final Logger logger = Logger.getLogger("edu.spl.reactor");
     private ProtocolTask<T> _task = null;

    private ConnectionHandler(SocketChannel sChannel, ReactorData<T> data, SelectionKey key) {
    public ConnectionHandler(DatagramSocketChannel dgChannel, ReactorData<T> data, SelectionKey key) {
        _sChannel = sChannel;
        _dgChannel = dgChannel;
        _data = data;        _skey = key;
        _protocol = _data.getProtocolMaker().create();
        _outData = new Vector< Pair<ByteBuffer,SocketAddress>>();
        _tokenizer = _data.getTokenizerMaker().create();
        _encoder = _data.getEncoderMaker().create();
    }

    private void initialize() {
        _skey.attach(this);
        _task = new ProtocolTask<T>(_protocol, _tokenizer, this);
    }

    public static <T> ConnectionHandler<T> create(
                SocketChannel sChannel, ReactorData<T> data, SelectionKey key) {
        ConnectionHandler<T> h = new ConnectionHandler<T>(sChannel, data, key);
        h.initialize();
        return h;
    }

    public synchronized void addOutData(ByteBuffer buf) {
    public synchronized void addOutData(T data, SocketAddress addr) {
```

```
    _outData.add(buf);
    _outData.add(new Pair<ByteBuffer,SocketAddress>(_encoder.decode(data),addr));
    switchToReadWriteMode();
}

private void closeConnection() {
    _skey.cancel();
    try {
        _sChannel.close();
        _dgChannel.close();
    } catch (IOException ignored) {
        ignored = null;
    }
}

public void read() {
    if (_protocol.shouldClose())
        return;

    SocketAddress address = _sChannel.socket().getRemoteSocketAddress();
    logger.info("Reading from " + address);

    ByteBuffer buf = ByteBuffer.allocate(BUFFER_SIZE);
    int numBytesRead = 0;
    SocketAddress address = null;
    while (true) {
    try {
        numBytesRead = _sChannel.read(buf);
        SocketAddress address = _dgChannel.receive(buf);

    } catch (IOException e) {
        numBytesRead = -1;
        address = null;
    }
   if (numBytesRead == -1) {
        logger.info("client on " + address + " has disconnected");
        closeConnection();
        _protocol.connectionTerminated();
        return;
    }
    if (addr == null)
       break;
    else {
        buf.flip();
       _task.addBytes(buf);
```

```
        ProtocolTask task = new ProtocolTask<T>(_protocol, _encoder.decode(buf),address, this);
        _data.getExecutor().execute(_task);
      }
    } //while
  }

  public synchronized void write() {
    if (_outData.size() == 0) {
      switchToReadOnlyMode();
       return;
    }
    while (true) {
      ByteBuffer buf = _outData.remove(0);
      if (buf.remaining() != 0) {
      int iSent = 0;
      try {
          _sChannel.write(buf);
          iSent =_sChannel.send(buf);
       } catch (IOException e) {
          e.printStackTrace();
          i = -1;
       }
       if (buf.remaining() != 0) {
       if (iSent <=0) {
         _outData.add(0, buf);
         break;
       }
    } //while
  }
  …
}

interface MessageEncoder<T> {
  T encode(ByteBuffer buf);
  ByteBuffer decode(T data);
}

class StringMessageEncoder implements MessageEncoder<StringMessage> {

   private final CharsetDecoder _decoder;
   private final CharsetEncoder _encoder;

  public StringMessageEncoder (Charset charset) {
    this._decoder = charset.newDecoder();
    this._encoder = charset.newEncoder();
```

```
  }

  public StringMessage  encode(ByteBuffer buf) {
        CharBuffer chars = CharBuffer.allocate(buf.remaining());
        decoder.decode(buf, chars, false);
        chars.flip();
        return new StringMessage(chars);
  }

  public T encode(StringMessage msg) {
    StringBuilder sb = new StringBuilder(msg.getMessage());
    return _encoder.encode(CharBuffer.wrap(sb));
  }
}
```

```
public class ProtocolTask<T> implements Runnable {

   private final ServerProtocol<T> _protocol;
   private final MessageTokenizer<T> _tokenizer;
   T _msg;
   SocketAddress _addr;
   private final ConnectionHandler<T> _handler;

   public ProtocolTask(ServerProtocol<T> protocol, MessageTokenizer<T> tokenizer, ConnectionHandler<T> h)
   public ProtocolTask(ServerProtocol<T> protocol, T msg, SocketAddress addr, ConnectionHandler<T> h)
  {
      this._protocol = protocol;
      this._tokenizer = tokenizer;
      this._msg = msg;
      this._addr = addr;
      this._handler = h;
   }

   public synchronized void run() {
     while (_tokenizer.hasMessage()) {
       T msg = _tokenizer.nextMessage();
       T response = this._protocol.processMessage(_msg);
       if (response != null) {
         try {
           ByteBuffer bytes = _tokenizer.getBytesForMessage(response);
           this._handler.addOutData(bytes);
           this._handler.addOutData(response,_addr);
         } catch (CharacterCodingException e) { e.printStackTrace(); }
       }
```

```
  }
}
```

סעיף ב (10 נקודות)

I יתרון: יעילות תקשורת. אלגוריתם האימות של UDP אינו שולח שוב הודעות
חסרון: הודעות שנשלחו עשויות שלא להגיע, או להגיע שלא על פי סדר

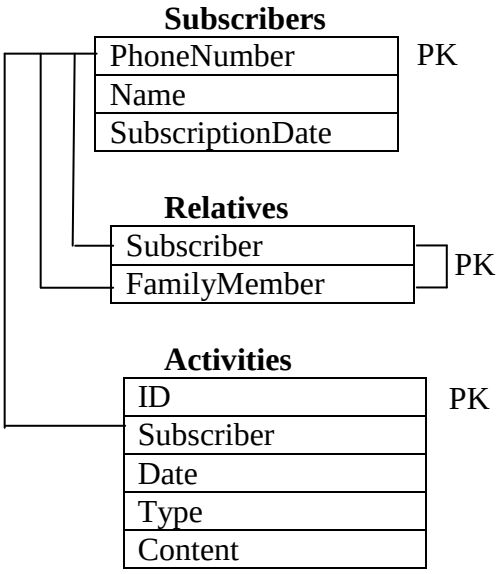
II TCP מגדיר מבוסס על קשר בין שני sockets, כך שלא ניתן לנהל דרך socket קשר עם כמה לקוחות.

III גם אם שכבת ה UDP ממומשת באופן מקבילי (דבר שאינו מוכרח כלל וכלל), בשכבת האפליקציה תבנית הריאקטור משתמשת בת'רד אחד לקריאה וכתיבה מ/ל הלקוח, כך שהגדרת מספר sockets לא מייעלת את התקשורת.
מי שהתייחס לבעיית גודל הבאפר בתקשורת עם מספר לקוחות דרך UDP socket אחד, קיבל את מלוא הנקודות.

שאלה 4 (10 נקודות)

סעיף א (5 נקודות)

ניתן מן הסתם לייצג את הנתונים לפתרון הבעיה המוצגת בדרכים שונות. נראה כאן מודל פשוט במיוחד, שבעזרתו ניתן להדגים את הנקודה העיקרית בשאילתה המבוקשת בסעיף הבא:



סעיף ב (5 נקודות)

```
SELECT Activities.Contents
   FROM (Subscribers JOIN Relatives
            ON Subscriber.PhoneNumber = Relatives.Subscriber)
         JOIN Activities
            ON Relatives.FamilyMember = Activities.Subscriber
   WHERE Subscribers.Name = 'Ringo Starr' AND
         Activities.Type = 'Message Received';
```

סרגל הבדיקה (לשני הסעיפים):

- קיבלנו פתרונות עם הנחות (סבירות) שונות, כולל פתרונות שהניחו שעלינו לשלוף הודעות בין בני המשפחה בלבד.
- על שיכפול נתוני לקוח בטבלת קרובי המשפחה הורדנו נקודה אחת.
- על ההנחה שבעזרת שם המשפחה ניתן לזהות את קרובי המשפחה הורדנו עד 3 נקודות (מה-10).
- ניסיון לייצג את כל קרובי המשפחה של לקוח כשדה יחיד בטבלת הלקוחות גרר הורדה של 3 נק'.
- אי ייצוג המשפחה כלל: 2- נק' מסעיף א'.
- שם לקוח כמפתח ראשי לטבלת הלקוחות: לא הורדנו.
- הפנייה מטבלת הלקוחות לטבלת הפעולות (הגורמת לשכפול פרטי לקוח עבור כל פעולה): 1-.
- קיבלנו גרסאות שונות המחברות בין חברי משפחה אחת בעזרת קוד מזהה.
- על שליפת הודעות של רינגו סטאר אך לא של קרובי משפחתו, עקב טעות בפרטי ה-JOIN הורדנו 2 נק'.
- שאילתה לשליפת הודעות ללא זיהוי רינגו סטאר ובני משפחתו: 4- נק'.

- שליפה ממספר טבלאות ללא ביצוע JOIN אלא בעזרת selection (תנאי WHERE): 3- נק'.
- שאילתה המתעלמת מטבלת הפעולות באופן שגם זיהוי בני המשפחה לא בא לידי ביטוי: 3- נק'.
- טעות יחסית תמימה ב-JOIN שפוגעת בנכונות, כגון פנייה לטבלה אחרת מהרצויה, גררה הורדת נקודה אחת.
- השמטת אחד מהתנאים (שהפעולה היא עבור הודעה או שהלקוח הינו רינגו סטאר) גררה הורדת חצי נקודה (תוך עיגול כלפי מעלה בציון הכולל, במידת הצורך).