The text in the image is a page from a document written in Hebrew. The document appears to be a test or assignment, possibly in computer science or a related field, as indicated by the text structure and content. The page includes instructions, descriptions of tasks, and some codes or algorithms. The text is written in a formal style, typical of academic or professional assignments.

The page includes a cover page with a date, name, and possibly a title or heading. The main content includes instructions for a task or exercise, which seems to involve concepts from computer science or programming.

The text is structured in paragraphs, likely explaining the task or providing instructions for completing a project. The use of Hebrew script indicates that the document is intended for a Hebrew-speaking audience.

Given the context and style, the document is likely an academic assignment, possibly for students learning computer science or a related discipline. The text contains instructions that seem designed to test or assess the student's understanding of specific concepts or skills.

Overall, the page is a formal academic document with a structured layout, typical of assignments or tests in a university or college setting.
interface Strategy {
    int next(int height);
}

class Relaxed implements Strategy {
    public int next(int height) {
        return height;
    }
}

class HurryUp implements Strategy {
    public int next(int height) {
        return height+1;
    }
}

class HurryDown implements Strategy {
    public int next(int height) {
        return height-1;
    }
}

interface Pedestrian {
    int getHeight();
    void setHeight(int height);
    Strategy getStrategy();
    void advance(StairCase staircase) throws InterruptedException;
}

class Passenger implements Pedestrian {
    private int _height;
    private final Strategy _strategy;

    Passenger(int height, Strategy strategy) { _height = height; _strategy = strategy; }
    public synchronized Strategy getStrategy() { return _strategy; }
    public synchronized int getHeight() { return _height; }
    public synchronized void setHeight(int height) { _height = height; }
    public synchronized void advance(StairCase staircase) throws InterruptedException {
        synchronized(staircase) {
            while (!((staircase != null &&
            staircase.getPedestrian(_strategy.next(getHeight())) - staircase.fromHeight()) == null &&
            (_height == staircase.fromHeight() || _height == staircase.toHeight() ||
            staircase.getPedestrian(_height - staircase.fromHeight()) == this)))
        }
    }
}
staircase.wait();
staircase.setPedestrian(null, _height - staircase.fromHeight());
_height = _strategy.next(_height);
staircase.setPedestrian(this, _height - staircase.fromHeight());
staircase.notifyAll();
}
}
}

interface StairCase {
  int fromHeight();
  int toHeight();
  Pedestrian getPedestrian(int i);
  void setPedestrian(Pedestrian pedestrian, int i);
  int size();
  int capacity();
}

class StairCaseImpl implements StairCase {
  private final Pedestrian[] _pedestrians;
  private final int _fromHeight;
  private final int _toHeight;

  StairCaseImpl(int fromHeight, int toHeight) throws Exception {
    if (toHeight - fromHeight < 0)
      throw new Exception("Negative staircase definition!");
    _fromHeight = fromHeight;
    _toHeight = toHeight;
    _pedestrians = new Pedestrian[_toHeight - _fromHeight + 1];
  }

  public int fromHeight () { return _fromHeight; }
  public int toHeight () { return _toHeight; }
  public int capacity() { return _pedestrians.length; }
  public synchronized int size() {
    int size=0;
    for (Pedestrian p : _pedestrians)
      if (p!=null)
        size++;
    return size;
  }
  public synchronized Pedestrian getPedestrian(int i) throws ArrayIndexOutOfBoundsException {
    return _pedestrians[i];
  }
}
public synchronized void setPedestrian(Pedestrian p, int i) throws ArrayIndexOutOfBoundsException {
    _pedestrians[i] = p;
}
}

class StairCaseMovementTask implements Runnable {
    private final StairCase _staircase;
    private final long _speed;
    StairCaseMovementTask(StairCase staircase, long speed) { _staircase = staircase; _speed = speed; }
    public void run() {
        while (true) {
            try {
                for (int i=_staircase.capacity() - 1; i>0; i--) {
                    _staircase.setPedestrian(_staircase.getPedestrian(i-1), i);
                    Pedestrian p = _staircase.getPedestrian(i);
                    if (p!=null) p.setHeight(p.getHeight()+1);
                }
                _staircase.setPedestrian(null, 0);
                synchronized(_staircase) { _staircase.notifyAll(); }
                Thread.sleep(_speed);
            } catch (Exception e) {}
        }
    }
}

class PedestrianMovementTask implements Runnable {
    private final Pedestrian _pedestrian;
    private final StairCase _stairCase;
    private final long _speed;
    PedestrianMovementTask(Pedestrian pedestrian, long speed, StairCase stairCase) {
        _pedestrian = pedestrian; _speed = speed; _stairCase = stairCase;
    }
    public void run() {
        while (true) {
            try {
                _pedestrian.advance(_stairCase);
                Thread.sleep(_speed);
            } catch (Exception e) {
                return;
            }
        }
    }
}
class Simulation {
    public static void main(String[] args) throws Exception {
        StairCase upStairCase = new StairCaseImpl(1, 39);
        Pedestrian pedestrian1 = new Passenger(1, new HurryUp());
        Pedestrian pedestrian2 = new Passenger(39, new HurryDown());
        new Thread(new StairCaseMovementTask(upStairCase, 1000)).start();
        new Thread(new PedestrianMovementTask(pedestrian1, 1000, upStairCase)).start();
        new Thread(new PedestrianMovementTask(pedestrian2, 1000, upStairCase)).start();
    }
}

A. The current implementation of the method advance in the class Passenger先进的方法先进于类

B. The implementation of the class Semaphore semaphore is not correct.

constructor summary

Semaphore(int permits)
Creates a Semaphore with the given number of permits.

method summary

void acquire()
Acquires a permit from this semaphore, blocking until one is available, or the thread
is interrupted.

void release()
Releases a permit, returning it to the semaphore.

boolean tryAcquire()
Acquires a permit from this semaphore, only if one is available at the time of
invocation.

A. The current implementation of the StairCaseImpl类的StairCaseImpl is safe under all parallel calculations, as it maintains the invariant defined for it. Run the simulation (i.e., run the parallel execution of the tasks StairCaseMovementTask and PedestrianMovementTask) incorrectly, with the removal of all synchronization.

B. Update the code so that it maintains the correctness of the parallel execution.

A. Run the simulation correctly. You should observe a deadlock.

B. Use the Semaphore Semaphore Semaphore, and update the implementation of the class Semaphore Semaphore, or return it to the semaphore. [10 points]

C. In addition to the two following questions, you can also answer question B.

C. In question A, we noticed that the class StairCaseImpl is safe under all parallel calculations, as it maintains the invariant defined for it. However, as mentioned above, the current implementation does not ensure that the concurrency of the methods (such as the method advance) is maintained. This creates a deadlock situation.

C. The current implementation is not correct, as it does not maintain the synchronization required for parallel execution. Therefore, the code needs to be updated to ensure correctness.

D. We can answer this question correctly as well. [10 points]
בון ערב המנהלים של "מדרגות נעות לישראל" הוחלطو את מערבות המדרגות הנעות של"ל וממשלו ב-
C++

```cpp
class StairCase {
public:
    virtual void setPedestrian(Pedestrian *pedestrian, int i) = 0;
    virtual Pedestrian *getPedestrian(int i) = 0;
    virtual int fromHeight() const = 0;
    virtual int toHeight() const = 0;
    virtual int capacity() const = 0;
    virtual ~StairCase();
};

class StairCaseImpl : public StairCase {
private:
    Pedestrian **_pedestrians;
    int _fromHeight;
    int _toHeight;
public:

    StairCaseImpl(int fromHeight, int toHeight): _fromHeight(fromHeight), _toHeight(toHeight) {
        _pedestrians = new Pedestrian*[_toHeight - _fromHeight + 1];
    }

    virtual ~StairCaseImpl() {
        delete [] _pedestrians;
    }

    StairCaseImpl(const StairCaseImpl& other): _fromHeight(other._fromHeight), _toHeight(other._toHeight),
    _pedestrians(other._pedestrians) {} 

    int fromHeight() const { return _fromHeight; }
    int toHeight() const { return _toHeight; }
    int capacity() const { return _toHeight - _fromHeight + 1; }
    Pedestrian *getPedestrian(int i) const {
        return _pedestrians[i];
    }
    void setPedestrian(Pedestrian *p, int i) {
        _pedestrians[i] = p;
    }
};
```

א. כתוב璇struments הבשנה נגדיר מנהיגי המדרגות, בואפר ילי, תוך בירזיה.
פומת אא המנהיגי היבérique יבאה אואר מעשיה אא בצלחת המדרגות (אא לשנת אא התומת הבדהה), יתנ ל_whitespace:

```cpp
void resize(int fromHeight, int toHeight, StairCaseImpl &staircase);
```
בגרם המדרגות המעודכן, יש למקם אתの人ים על המדרגות אחריה תוךنشرי ליו מיקומים מקודרים (.= הנגב. שלמה
לא משנתה). אם מיקומים חודר אנטיולית המדרגות התורמים, يتمulner על המדרגות.8 [عكسית]
ב. כדי לבודק את הנגזרה הש -=, נכתבה התוכנית הבאה:

c. כדי להס sca כי הקמודים שנמצאים על המדרגות, resize

d. כדי לבודק את התוכנית באופן מקודך 12 [عكسית] .foo ל here1

ב. הזרת התוכנית נבדל ישורע להיקלעל לשנייה ויכוח במשפט.

כנוה נפעלה, תוכן ואית המתלהקה [6 [عكسית] StairCaseImpl

d. מתוכנתה, בגרמת הקמוד, עבודה (על פי תכניתה גורם מהגרה עצמית) ואת המדרגות. כפג께

ב main() מ test2. ושינה זה גורם לבלית קומפילציה. ח voks על הקمد הקמד שלחביתית תיפות.4 [عكسית]

void main() {
   ...
   StairCase * test2 = new StairCaseImpl(fromHeight, toHeight);
   foo(test2, fromHeight*2, toHeight);
}
במקרה בו מוכחת ניסוח פעולות מרחוקה_BTמדרגות – כולה ביצוע המשימה במקלט אחד במקלט אחרים התפקיד של המדרגות, באתרים שונים. - StairCaseMovementTask

```java
public class StairCaseControl {
    public static void main(String[] args) {
        try {
            StairCase upStairCase = (StairCase) Naming.lookup("132.87.45.3:4004/StairCase1");
            new Thread(new StairCaseMovementTask(upStairCase, 1000)).start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

לשם כף המדרגות על המדרגות, כ-Remote

```java
interface StairCase extends java.rmi.Remote {
    int fromHeight() throws java.rmi.RemoteException;
    int toHeight() throws java.rmi.RemoteException;
    Pedestrian getPedestrian(int i) throws java.rmi.RemoteException;
    void setPedestrian(Pedestrian pedestrian, int i) throws java.rmi.RemoteException;
    int size() throws java.rmi.RemoteException;
    int capacity() throws java.rmi.RemoteException;
}
```

class StairCaseImpl extends java.rmi.server.UnicastRemoteObject implements StairCase {
    ...
}

מכהלה מפרסמת ב-Simulation המחלקה_BTמדרגות מכף製作 המדרגות. א-אינה מפרסמת ת'רד מעל הCLS_Simulation (משימה זו מבוצעת על ידי ת'רד בתהליך StairCaseControl.)

```java
class Simulation {
    public static void main(String[] args) throws Exception {
        StairCase upStairCase = new StairCaseImpl(1, 39);
        Naming.rebind("132.87.45.3:4004/StairCase1", upStairCase);
        Pedestrian pedestrian1 = new Passenger(1, new HurryUp());
        Pedestrian pedestrian2 = new Passenger(39, new HurryDown());
        new Thread(new PedestrianMovementTask(pedestrian1, 1000, upStairCase)).start();
    }
}
```
A. In the method Simulation, three more print statements are added, printing the height of pedestrian 1, and the content of the first staircase (where it is located at the beginning of the simulation).

Explain how it is possible (even after the required correction in question A) that the first print statement prints a height of 1 for pedestrian, but he is not located in the first staircase (index 2 of the stairs starting at height 1) – namely, the second print statement prints null while the third print statement prints pedestrian 1.

5 points

To tackle the problem of question A, Pedestrian was declared Remote interface

```java
interface Pedestrian extends java.rmi.Remote {
    int getHeight() throws java.rmi.RemoteException;
    void setHeight(int height) throws java.rmi.RemoteException;
    Strategy getStrategy() throws java.rmi.RemoteException;
    void advance(StairCase stairCase) throws java.rmi.RemoteException;
}
```

```java
class Passenger implements Pedestrian
    extends java.rmi.server.UnicastRemoteException implements Pedestrian {
    ...
}
```

B. Server methods use the thread for access, not a. 5 points

C. In order to reduce communication operations, it was decided not to redefine the Pedestrian interface (and its implementation (Passenger) as Serializable (in the Remote context) and the correct way to use StairCaseMovementTask.

Update the code according to this proposal, and indicate whether the simulation given does indeed save communication operations.

12 points

It is possible to answer the next question even if you did not answer the previous questions.

D. It is given that the communication stub of StairCase, with the Stub, is based on the Reactor template. In the original code of the Reactor, the run() method in the ProtocolTask is synchronized with the Stub.

The student suggested that in order to improve, first under synchronization, to collect the messages that arrived in a local list. Then release the synchronization (in order to allow other threads to work on the ProtocolTask,) and then execute the messages one after another.

[5 points]
class ProtocolTask implements Runnable {
    private final ServerProtocol _protocol;
    private final StringMessageTokenizer _tokenizer;
    private final ConnectionHandler _handle;
    /* The fifo queue, which holds data coming from the socket. Access to the queue is serialized, to ensure correct
     * processing order - even if more data is received by the reactor while previous data is still being processed.*
    private final Vector<ByteBuffer> _buffers = new Vector<ByteBuffer>();
    ...
    public synchronized void run() {
        // first, add all the bytes we have to the tokenizer
        synchronized (_buffers) {
            while (_buffers.size() > 0) {
                ByteBuffer buf = _buffers.remove(0);
                this._tokenizer.addBytes(buf);
            }
        }
        // now, go over all complete messages and add the to the local 'messages' list
        List<String> messages = new ArrayList<String>();
        synchronized (this) {
            while (_tokenizer.hasMoreMessages())
                messages.add(_tokenizer.nextMessage());
        }
        // process all messages of the local 'messages' list
        for (String message : messages) {
            String response = this._protocol.processMessage(msg);
            if (response != null) {
                try {
                    ByteBuffer bytes = _tokenizer.getBytesForMessage(response);
                    this._handler.addOutData(bytes);
                } catch (CharacterCodingException e) { e.printStackTrace(); }
            }
        }
    }
}
במקרה של גרם מדרגות הכולל הפרטים הבאים בטבלאות הבאות:

- מיקום גרם המדרגות (שם הארץ, שם העיר, שם המתחם)
- אוריינטציה (עולה או יורד)
- רישום של החופをしている המועלים
-гер
- מספר העובדים בו
- תאריך

ב. כעת נדרש להוסיף צוות של עובדים. הנתון היחיד עבור עובד הם שמו (ניתן להניח כי שמו של העובד מזהה אותו באופן ייחודי).

הוסף את העובדים למודל הנתונים

א. צעדים נוספים למסכים לכל גרם מדרגות צועד של עובד. הנחט᾿ת היהに向ד, עובד יועבר overlapping שמע (ňёнן לחה יי).

 gammilla שאלת SQL המתייחסת לשם העובדים בגרם המדרגות הממוקם במקים

Forth Rail "Bridge ב. הגדרו שאלת SQL המתייחסת לשם העובדים בגרם המדרגות הממוקם במקים

| Create table Location ( |
| LocationId int primary key, |
| LocationName varchar(200), |
| City varchar(200), |
| Country varchar(100)) |

| Create table Staircase ( |
| StaircaseId int Primary Key, |
| LocationId int Foreign Key references Location, |
| Orientation int // 1 = ascending, 2 = descending |

| Create table StaircaseOperation ( |
| StaircaseId int Foreign Key references Staircase, |
| OperationDay datetime, |
| PassengersNumber int, |
| AverageTripDuration int, // average duration in milliseconds |
| Primary Key (StaircaseId, OperationDay) |

שאלה 4

(10 נקודות)
public class ConnectionHandler {
    protected final SocketChannel _sChannel;
    protected final ReactorData _data;
    protected final AsyncServerProtocol _protocol;
    protected final StringMessageTokenizer _tokenizer;
    protected Vector<ByteBuffer> _outData;
    protected final SelectionKey _skey;
    private ProtocolTask _task;
    
    // Post data in the pending data queue, so that the connectionHandler will send it through the socket.
    // switchToReadWriteMode() subscribes this handler key to the OP_WRITE event
    // This event will immediately fire because the output buffer of the channel is empty.
    // It will keep firing as long as the output buffer is not filled.
    // When we are done sending pending data, we will unsubscribe from OP_WRITE.
    public synchronized void addOutData(ByteBuffer buf) {
        _outData.add(buf);
        switchToReadWriteMode();
    }
    
    // Reads incoming data from the client. Reads some bytes from the SocketChannel
    // Create a protocolTask, to process this data, possibly generating an answer.
    // Inserts the Task to the ThreadPool
    public void read() {
        // Do not read if protocol has terminated. Only write of pending data is
        // allowed when the protocol asked to close the connection.
        if (_protocol.shouldClose())
            return;
        
        SocketAddress address = _sChannel.socket().getRemoteSocketAddress();
        logger.info("Reading from " + address);
        ByteBuffer buf = ByteBuffer.allocate(BUFFER_SIZE);
        int numBytesRead = 0;
        try {
            numBytesRead = _sChannel.read(buf);
        } catch (IOException e) {
            numBytesRead = -1;
        }
        
        if (numBytesRead == -1) {  // Is the channel closed?
            // No more bytes can be read from the channel
            logger.info("client on " + address + " has disconnected");
            closeConnection();
            // tell the protocol that the connection terminated.
            _protocol.connectionTerminated();
        }
    }
}
return;

// Add the buffer to the protocol task
buf.flip();
_task.addBytes(buf);
// Add the protocol task to the reactor which will parse and process the data
// when a thread becomes available for it.
_data.getExecutor().execute(_task);

// Attempts to send data to the client. If all the data has been successfully sent, the ConnectionHandler will
// automatically switch to read only mode, otherwise it will stay in its current mode (which is read / write).
public synchronized void write() {
    if (_outData.size() == 0) { // if nothing left in the output string, go back to read mode
        switchToReadOnlyMode();
        return;
    }
    // If there is something to send - send the first byte buffer
    // We will return to this write() operation very soon because the selector will keep firing the OP_WRITE event
    // after we are done writing this buffer and check if there are more buffers to be sent.
    ByteBuffer buf = _outData.remove(0);
    if (buf.remaining() != 0) {
        // Check if the buffer contains more data: we could not send all of the buffer in one write
        // (the output buffer of the socket got full). So we remember that there is more data to be sent.
        // We will receive a new OP_WRITE event when the output buffer of the socket
        // is not full anymore and complete the write operation then.
        _sChannel.write(buf);
        if (buf.remaining() != 0)
            _outData.add(0, buf);
    }
    // Check if the protocol asked us to close this connection.
    // If it did, we remain open as long as there are pending data to be sent.
    // As soon as all the data has been sent, we can close the connection.
    if (_protocol.shouldClose()) {
        switchToWriteOnlyMode();
        if (buf.remaining() == 0) {
            logger.info("disconnecting client on " + _sChannel.socket().getRemoteSocketAddress());
            closeConnection();
        }
    }
}