The **LightBoard** class models a two-dimensional display of lights, where each light is either on or off, as represented by a Boolean value. You will implement a constructor to initialize the display and a method to evaluate a light.

```java
public class LightBoard {
    /** The lights on the board, where true represents on and false represents off. */
    private boolean[][] lights;

    /** Constructs a LightBoard object having numRows rows and numCols columns.
     * Precondition: numRows > 0, numCols > 0
     * Postcondition: each light has a 40% probability of being set to on. */
    public LightBoard(int numRows, int numCols)
    { /* to be implemented in part (a) */ }

    /** Evaluates a light in row index row and column index col and returns a status
     * as described in part (b).
     * Precondition: row and col are valid indexes in lights. */
    public boolean evaluateLight(int row, int col)
    { /* to be implemented in part (b) */ }

    // There may be additional instance variables, constructors, and methods not shown.
}
```

GO ON TO THE NEXT PAGE.
AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTION

(a) Write the constructor for the LightBoard class, which initializes lights so that each light is set to on with a 40% probability. The notation lights[r][c] represents the array element at row r and column c.

Complete the LightBoard constructor below.

/**  Constructs a LightBoard object having numRows rows and numCols columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols)
Write the method `evaluateLight`, which computes and returns the status of a light at a given row and column based on the following rules.

1. If the light is on, return `false` if the number of lights in its column that are on is even, including the current light.
2. If the light is off, return `true` if the number of lights in its column that are on is divisible by three.
3. Otherwise, return the light’s current status.

For example, suppose that `LightBoard sim = new LightBoard(7, 5)` creates a light board with the initial state shown below, where `true` represents a light that is on and `false` represents a light that is off. Lights that are off are shaded.

```
lights

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
```

Sample calls to `evaluateLight` are shown below.

<table>
<thead>
<tr>
<th>Call to <code>evaluateLight</code></th>
<th>Value Returned</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sim.evaluateLight(0, 3);</code></td>
<td><code>false</code></td>
<td>The light is on, and the number of lights that are on in its column is even.</td>
</tr>
<tr>
<td><code>sim.evaluateLight(6, 0);</code></td>
<td><code>true</code></td>
<td>The light is off, and the number of lights that are on in its column is divisible by 3.</td>
</tr>
<tr>
<td><code>sim.evaluateLight(4, 1);</code></td>
<td><code>false</code></td>
<td>Returns the light’s current status.</td>
</tr>
<tr>
<td><code>sim.evaluateLight(5, 4);</code></td>
<td><code>true</code></td>
<td>Returns the light’s current status.</td>
</tr>
</tbody>
</table>
Complete the `evaluateLight` method below.

```java
/** Evaluates a light in row index row and column index col and returns a status
 * as described in part (b).
 * Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
```