

Name _____ Period _____

1. 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.

```
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 60% 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.
}
```

(a) Write the constructor for the `LightBoard` class, which initializes lights so that each light is set to on with a 60% 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 60% probability of being set to on.  
 */
```

```
public LightBoard(int numRows, int numCols) {
```

```
}
```

/4

(b) 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 off, return false if the number of lights in its column that are on is odd, including the current light.
2. If the light is on, 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.

Class information for this question

```
public class LightBoard
private boolean[][] lights
public LightBoard(int numRows, int numCols)
public boolean evaluateLight(int row, int col)
```

Complete the `evaluateLight` method below.

```
/** 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) {
```

```
}
```

/5

2.

This question involves reasoning about arrays of integers. You will write two static methods, both of which are in a class named `ArrayTester`.

```
public class ArrayTester
{
    /** Returns an array containing the elements of column c of arr2D in the same order as
     * they appear in arr2D.
     * Precondition: c is a valid column index in arr2D.
     * Postcondition: arr2D is unchanged.
     */
    public static int[] getColumn(int[][] arr2D, int c)
    { /* to be implemented in part (a) */ }

    /** Returns true if and only if every value in arr1 appears in arr2.
     * Precondition: arr1 and arr2 have the same length.
     * Postcondition: arr1 and arr2 are unchanged.
     */
    public static boolean hasAllValues(int[] arr1, int[] arr2)
    { /* implementation not shown */ }

    /** Returns true if arr contains any duplicate values;
     * false otherwise.
     */
    public static boolean containsDuplicates(int[] arr)
    { /* implementation not shown */ }

    /** Returns true if square is a Latin square as described in part (b);
     * false otherwise.
     * Precondition: square has an equal number of rows and columns.
     * square has at least one row.
     */
    public static boolean isLatin(int[][] square)
    { /* to be implemented in part (b) */ }
}
```

- (a) Write a static method `getColumn`, which returns a one-dimensional array containing the elements of a single column in a two-dimensional array. The elements in the returned array should be in the same order as they appear in the given column. The notation `arr2D[r][c]` represents the array element at row `r` and column `c`.

The following code segment initializes an array and calls the `getColumn` method.

```
int[] [] arr2D = { { 0, 1, 2 },
                  { 3, 4, 5 },
                  { 6, 7, 8 },
                  { 9, 5, 3 } };

int[] result = ArrayTester.getColumn(arr2D, 1);
```

When the code segment has completed execution, the variable `result` will have the following contents.

`result: {1, 4, 7, 5}`

Complete method `getColumn` below.

```
/** Returns an array containing the elements of column c of arr2D in the same order as they
 * appear in arr2D.
 * Precondition: c is a valid column index in arr2D.
 * Postcondition: arr2D is unchanged.
 */
public static int[] getColumn(int[] [] arr2D, int c)
```

- (b) Write the static method `isLatin`, which returns `true` if a given two-dimensional square array is a *Latin square*, and otherwise, returns `false`.

A two-dimensional square array of integers is a Latin square if the following conditions are true.

- The first row has no duplicate values.
- All values in the first row of the square appear in each row of the square.
- All values in the first row of the square appear in each column of the square.

Examples of Latin Squares

1	2	3
2	3	1
3	1	2

10	30	20	0
0	20	30	10
30	0	10	20
20	10	0	30

Examples that are NOT Latin Squares

1	2	1
2	1	1
1	1	2

Not a Latin square
because the first row
contains duplicate
values

1	2	3
3	1	2
7	8	9

Not a Latin square
because the elements of
the first row do not all
appear in the third row

1	2
1	2

Not a Latin square
because the elements of
the first row do not all
appear in either column

The `ArrayTester` class provides two helper methods: `containsDuplicates` and `hasAllValues`. The method `containsDuplicates` returns `true` if the given one-dimensional array `arr` contains any duplicate values and `false` otherwise. The method `hasAllValues` returns `true` if and only if every value in `arr1` appears in `arr2`. You do not need to write the code for these methods.

Complete method `isLatin` below. Assume that `getColumn` works as specified, regardless of what you wrote in part (a). You must use `getColumn`, `hasAllValues`, and `containsDuplicates` appropriately to receive full credit.

```
/** Returns true if square is a Latin square as described in part (b);
 *     false otherwise.
 *   Precondition: square has an equal number of rows and columns.
 *                   square has at least one row.
 */
public static boolean isLatin(int [] [] square)
```

