4. This question involves reasoning about a two-dimensional (2D) array of integers. You will write two static methods, both of which are in a single enclosing class named `Successors` (not shown). These methods process a 2D integer array that contains consecutive values. Each of these integers may be in any position in the 2D integer array. For example, the following 2D integer array with 3 rows and 4 columns contains the integers 5 through 16, inclusive.

2D Integer Array

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 15 | 5 | 9 | 10 |
| 1 | 12 | 16 | 11 | 6 |
| 2 | 14 | 8 | 13 | 7 |

The following `Position` class is used to represent positions in the integer array. The notation `(r,c)` will be used to refer to a `Position` object with row `r` and column `c`.

```
public class Position
{
    /** Constructs a Position object with row r and column c. */
    public Position(int r, int c)
    {  /* implementation not shown */  }

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

(a) Write a `static` method `findPosition` that takes an integer value and a 2D integer array and returns the position of the integer in the given 2D integer array. If the integer is not an element of the 2D integer array, the method returns `null`.

For example, assume that array `arr` is the 2D integer array shown at the beginning of the question.

- The call `findPosition(8, arr)` would return the `Position` object `(2,1)` because the value `8` appears in `arr` at row 2 and column 1.
- The call `findPosition(17, arr)` would return `null` because the value `17` does not appear in `arr`.

Complete method `findPosition` below.

```
/** Returns the position of num in intArr;
 *  returns null if no such element exists in intArr.
 *  Precondition: intArr contains at least one row.
 */
public static Position findPosition(int num, int[][] intArr)
```

Part (b) begins on page 18.

# AP® COMPUTER SCIENCE A
# 2017 SCORING GUIDELINES

## Question 4: Successor Array

| **Part (a)** | `findPosition` | **5 points** |
|---|---|---|

**Intent:** *Find the position of a given integer in a 2D integer array*

**+1** Accesses all necessary elements of `intArr` (*no bounds errors*)

**+1** Identifies `intArr` element equal to `num` (*in context of an* `intArr` *traversal*)

**+1** Constructs `Position` object with same row and column as identified `intArr` element

**+1** Selects constructed object when `intArr` element identified; `null` when not

**+1** Returns selected value

| **Part (b)** | `getSuccessorArray` | **4 points** |
|---|---|---|

**Intent:** *Create a successor array based on a 2D integer array*

**+1** Creates 2D array of `Position` objects with same dimensions as `intArr`

**+1** Assigns a value to a location in 2D successor array using a valid call to `findPosition`

**+1** Determines the successor `Position` of an `intArr` element accessed by row and column (*in context of* `intArr` *traversal*)

**+1** Assigns all necessary locations in successor array with corresponding position object or `null` (*no bounds errors*)

**Question-Specific Penalties**

**-1** (s) Uses confused identifier `Arr`

**-1** (t) Uses `intArr[].length` as the number of columns

**-1** (u) Uses non-existent accessor methods from `Position`

# AP® COMPUTER SCIENCE A
# 2017 SCORING GUIDELINES

## Question 4: Scoring Notes

| **Part (a)** `findPosition` | | | **5 points** |
|---|---|---|---|
| Points | Rubric Criteria | Responses earn the point if they ... | Responses will not earn the point if they ... |
| +1 | Accesses all necessary elements of `intArr` (*no bounds errors*) | | • use `if (...) return; else return null;` inside loop<br>• confuse row and column bounds<br>• fail to traverse `intArr` |
| +1 | Identifies `intArr` element equal to `num` (*in context of an `intArr` traversal*) | | • use `.equals` instead of `==` |
| +1 | Constructs `Position` object with same row and column as identified `intArr` element | | • omit keyword `new`<br>• use `(r,c)` instead of `Position(r,c)` |
| +1 | Selects constructed object when `intArr` element identified; `null` when not | • use `"null"` instead of `null`<br>• construct a `String` object using row and column indices | • use `if (...) return; else return null;` inside loop<br>• use `(r,c)` instead of `Position(r,c)` |
| +1 | Returns selected value | | |
| **Part (b)** `getSuccessorArray` | | | **4 points** |
| Points | Rubric Criteria | Responses earn the point if they ... | Responses will not earn the point if they ... |
| +1 | Creates 2D array of `Position` objects with same dimensions as `intArr` | | • omit keyword `new` |
| +1 | Assigns a value to a location in 2D successor array using a valid call to `findPosition` | • call `Successors.findPosition(…)` | • reimplement the code from `findPosition`<br>• call `findPosition` with a single argument<br>• call `this.findPosition(…)` |
| +1 | Determines the successor `Position` of an `intArr` element accessed by row and column *(in context of `intArr` traversal*) | • reimplement the code from `findPosition` | • call `findPosition` using an integer that is not identified with a location in `intArr`<br>• call `findPosition` with a single argument |
| +1 | Assigns all necessary locations in successor array with corresponding position object or `null` (*no bounds errors*) | • use `SuccessorArray` dimensions correctly, even if `SuccessorArray` was not initialized properly<br>• only assign non-null entries to `SuccessorArray` | • reimplement the code from `findPosition` but mishandle the null case.<br>• fail to traverse `intArr` |

Return is not assessed in Part (b).

# AP® COMPUTER SCIENCE A
# 2017 SCORING GUIDELINES

## Question 4: Successor Array

*Part (a)*

```
public static Position findPosition(int num, int[][] intArr)
{
      for (int row=0; row < intArr.length; row++)
      {
            for (int col=0; col < intArr[0].length; col++)
            {
                  if (intArr[row][col] == num)
                  {
                        return new Position(row, col);
                  }
            }
      }
      return null;
}
```

*Part (b)*

```
public static Position[][] getSuccessorArray(int[][] intArr)
{
      Position[][] newArr = new Position[intArr.length][intArr[0].length];

      for (int row=0; row < intArr.length; row++)
      {
            for (int col=0; col < intArr[0].length; col++)
            {
                  newArr[row][col] = findPosition(intArr[row][col]+1, intArr);
            }
      }
      return newArr;
}
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.