Day 11: 2D Arrays | HackerRank

Terms you'll find helpful in completing today's challenge are outlined below, along with sample Java code (where appropriate).

Also known as *multidimensional* arrays, they are very similar to the regular *1D Array* data structure we've already discussed.

Consider the following code:

```
int rowSize = 2;
int colSize = 4;
int[][] myArray = new int[rowSize][colSize];
```

This creates a 2×4 matrix where each element, (i, j), can be graphically represented as follows:

```
(0, 0) (0, 1) (0, 2) (0, 3)
(1, 0) (1, 1) (1, 2) (1, 3)
```

You may find it helpful to think of these (i, j) elements in terms of real-world structures such as the cells in a spreadsheet table.

To fill the array's cells with values, you can use a *nested loop*. The outer loop represents the matrix's *rows* and uses i as its variable, and the inner loop represents the matrix's *columns* and uses j as its variable. The code below assigns the value of $count$ to each element in the *2D* array we declared previously:

```
int count = 0;

for(int i = 0; i < rowSize; i++) {

    for(int j = 0; j < colSize; j++, count++) {
        myArray[i][j] = count;
    }
}
```

If we print the contents of each row:

```
for(int i = 0; i < rowSize; i++) {

    // print the row of space-separated values
    for(int j = 0; j < colSize; j++) {
        System.out.print(myArray[i][j] + " ");
    }
    // end of row is reached, print newline
    System.out.println();
}
```

we'll see the following output:

Additional Language Resources

C++ Arrays (1D and 2D)
Python Nested List Comprehensions

Solve Problem