COMP1006/1406 – Summer 2018

Instructions for submitting will be posted to cuLearn.

In assignment specifications, text appearing `like this` represents code in a program or things that a user types when running a program. Text that appears `like this` represents output from your program or methods.

1: Arrays

➡ In the provided `MinDistance.java` file, complete the `distance` method. For a given target value, the method returns the minimal distance between any two instances of the target in the input array of numbers.

The distance between two numbers in the array is defined to be the number of array cells between them. Consider the following array of numbers

5	3	7	2	3	4	4	7	3	-1	3	4	7	5

The distance between the two 5's in the array is 12. The distance between the first two 4's in the array is 0. The *minimal* distance between any two 3's is 1. The minimal distance between any two 7's is 4.

```
public static int distance(byte target, byte[] numbers){
  /* purpose: determine the minimal distance between any
   *          two instances of target in the numbers array
   * input  : target is the number we are looking for
   *          numbers is an array of numbers
   * output : the method returns
   *           o) the minimal distance between any two instances
   *              of target in the numbers array, if there are
   *              at least two instances of target present, or
   *           o) -2 if target is not in the numbers array, or
   *           o) -1 if target appears only once in the numbers array
   */
```

Examples: Let N denote the array shown above.

`MinDistance.distance( (byte)5, N)` returns `12`

`MinDistance.distance( (byte)4, N)` returns `0`

`MinDistance.distance( (byte)7, N)` returns `4`

`MinDistance.distance( (byte)3, N)` returns `1`

`MinDistance.distance( (byte)2, N)` returns `-1`

`MinDistance.distance( (byte)13, N)` returns `-2`

2: Sudoku Checker

➡ In the provided file `SudokuChecker.java`, complete the following methods:

```
public static boolean checkRow(int row, byte[][] grid)
public static boolean checkColumn(int column, byte[][] grid)
public static boolean checkSubregion(int region, byte[][] grid)
```

These methods are used in the provided `check` method that takes a 2-dimensional (9×9) array of numbers and determines if it represents a valid (solved) sudoku grid of numbers. A valid sudoku grid has the following properties:

- each row contains all of the digits from 1 to 9,
- each column contains all of the digits from 1 to 9,
- each of the nine 2-dimensional 3×3 subgregions contain all of the digits 1 to 9.

For example, the following is a valid sudoku grid (with $n = 3$). The nine subregions are shown shaded.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

The rows are labelled 0 to 8 (from top to bottom), the columns are labelled 0 to 8 (from left to right), and the subregions are labelled 0 to 8 as follows:

0	1	2
3	4	5
6	7	8

Note: The `check` method will not be tested. We will be testing the three other methods with the auto-testing.

Submission Recap

A complete assignment will consist of a single file (`assignment1.zip`) with the following two (2) files included: `MinDistance.java` and `SudokuChecker.java`.