CS 61B
Summer 2020

Small Group Tutoring
Section 4: Arrays [Solution]

Worksheet 4

1 Read Me

Describe what each of the following methods does. You may assume that `values` contains at least one element.

```
private static boolean method1 (int[] values) {
    int k = 0;
    while (k < values.length - 1) {
        if (values[k] > values[k+1]) {
            return false;
        }
        k = k + 1;
    }
    return true;
}
```

Solution: `method1` returns true if `values` is non-decreasing, i.e. if each value in `values` is larger than *or equal to* the previous element.

```
private static void method2 (int[] values) {
    int k = 0;
    while (k < values.length / 2) {
        int temp = values[k];
        values[k] = values[values.length - 1 - k];
        values[values.length - 1 - k] = temp;
        k = k + 1;
    }
}
```

Solution: `method2` reverses `values` in place. Note that method2 has no return value and instead mutates `values`.

2 Flatten

Write a method flatten that takes in a 2-D int array x and returns a 1-D int array that contains all of the arrays in x concatenated together. For example, flatten({{1, 3, 7}, {}, {9}}) should return {1, 3, 7, 9}.

Solution:

```
public static int[] flatten(int[][] x) {
    //newArraySize will hold the length of the flattened list
    int newArraySize = 0;

    for (int i = 0; i < x.length; i+=1) {
        //calculating the length of flattened list
        newArraySize += x[i].length;
    }
    int[] newArray = new int[newArraySize];

    //newArrayIndex will be the index used to access the flattened list
    int newArrayIndex = 0;

    for (int i = 0; i < x.length; i+=1) {
        for (int j = 0; j < x[i].length; j+=1) {

            /* index into the flattened list using newArrayIndex
            and store the element from the original
            2D-array at position (i, j) */

            newArray[newArrayIndex] = x[i][j];

            /* increment the newArrayIndex for next time
            (next position in the flattened array) */

            newArrayIndex += 1;
        }
    }
    return newArray;
}
```