Problem 1

Write a static function **trans** which takes a two-dimensional *rectangular* (by assumption) array of **int**s. The function creates and returns a new two-dimensional array the consecutive *columns* of which contain elements from consecutive *rows* of the original array. It doesn't print anything!

For example, the following program

download *Arr2DTr.java*

```
import java.util.Arrays;
public class Arr2DTr {
    // ...
    public static void main (String[] args) {
        int[][] a = { {1,2,3,4,5,6},
                      {2,3,4,5,6,7},
                      {3,4,5,6,7,8},
                      {4,5,6,7,8,9} };
        for (int[] r : a)
            System.out.println(Arrays.toString(r));
        System.out.println();
        for (int[] r : trans(a))
            System.out.println(Arrays.toString(r));
    }
}
```

should print

```
[1, 2, 3, 4, 5, 6]
[2, 3, 4, 5, 6, 7]
[3, 4, 5, 6, 7, 8]
[4, 5, 6, 7, 8, 9]

[1, 2, 3, 4]
[2, 3, 4, 5]
[3, 4, 5, 6]
[4, 5, 6, 7]
[5, 6, 7, 8]
[6, 7, 8, 9]
```

Problem 2

Create and test functions taking a two-dimensional array of **double**s and returning a **boolean**[1]

[1] Functions returning a **boolean** are called *predicates*.

- **isRectangular** — checking if the given array is rectangular;
- **isSquare** — checking if the given array is square;
- **isSymmetric** — checking if the given array is symmetric (i.e., is square and $a_{ij} = a_{ji}$).

Create also predicates taking *two* two-dimensional arrays

- **isSameShape** — checking if given arrays are of the same shape, i.e., all dimensions are equal;
- **isSame** — checking if given arrays are identical.

Problem 3

Define a two-dimensional array of **String**s representing the results of matches in a tournament, for example as below

download *Arr2DTournament.java*

```
String[][] arr =
    { {"Germany",  "2", "Scotland", "1"},
      {"Poland",   "2", "Germany",  "0"},
      {"Germany",  "1", "Ireland",  "1"},
      {"Poland",   "2", "Scotland", "2"},
      {"Scotland", "1", "Ireland",  "0"},
      {"Ireland",  "1", "Poland",   "1"},
      {"Ireland",  "1", "Scotland", "1"},
      {"Germany",  "3", "Poland",   "1"},
      {"Scotland", "2", "Germany",  "3"},
      {"Ireland",  "1", "Germany",  "0"},
      {"Scotland", "2", "Poland",   "2"},
      {"Poland",   "2", "Ireland",  "1"} };
```

and a four-element array of **int**s representing scores of teams of Germany, Ireland, Poland and Scotland (in this order). The program calculates total score for each team (3 points for a win, 1 for a draw, 0 for a defeat), puts them into the array and then prints it.

It will be very helpful to define a small function which, given the name of a country, returns its index in the array of total scores (**switch** expression would be appropriate here).

If a string `str` represents a number, like `"435"`, you can get this number as an **int** using:

```
int n = Integer.parseInt(str);
```

For data as above you should get scores $[10, 6, 9, 6]$.

Problem 4

Write a function **selSort** which sorts a given array of **int**s (orders its elements from the smallest to the greatest) using *selection sort* algorithm. Namely, we iterate over positions (indices) of the array from index 0 to the last but one and for each index (say, `i`) we find the *index* of the smallest element among those with indices larger

than i (i.e., to the right of the i-th element). If the element found is smaller than the i-th, we swap them. Note that the number of swaps will be at most $n-1$ where n is the size of the array (see Problem **??**).

Write two static functions, both taking a *two-dimensional* array of **int**s, which

- **sortRows** — sorts separately each "row" of the array (note that the array need not be rectangular);
- **sortCols** — sorts separately each "column" of the array (this, of course, can be done only for rectangular arrays).

In both cases you can use previously defined **SelSort** function, although in the second case it would be better to implement sorting inside the **SortCols** function.

Write also a function which prints a two-dimensional array on the console.

For example, the following program

download *Arr2DSelSort.java*

```
public class Arr2DSelSort {
    public static void selSort(int[] arr) { ... }
    public static void sortRows(int[][] arr) { ... }
    public static void sortCols(int[][] arr) { ... }
    public static void printArr2D(int[][] arr) { ... }

    public static void main (String[] args) {
       int[][] a = { {3,2,6,1,3,5,6,1,3},
                     {3,1,2,1,5,7,2},
                     {8,9,2,1} };
       System.out.println("Before:");
       printArr2D(a);
       sortRows(a);
       System.out.println("After:");
       printArr2D(a);

       int[][] b = { {3,2,6,1,6},
                     {7,1,2,1,5},
                     {5,3,1,8,7},
                     {8,9,2,7,1} };
       System.out.println("Now columns\nBefore:");
       printArr2D(b);
       sortCols(b);
       System.out.println("After:");
       printArr2D(b);
    }
}
```

should print

```
[3, 2, 6, 1, 3, 5, 6, 1, 3]
[3, 1, 2, 1, 5, 7, 2]
[8, 9, 2, 1]
After:
[1, 1, 2, 3, 3, 3, 5, 6, 6]
[1, 1, 2, 2, 3, 5, 7]
[1, 2, 8, 9]
Now columns
Before:
[3, 2, 6, 1, 6]
[7, 1, 2, 1, 5]
[5, 3, 1, 8, 7]
[8, 9, 2, 7, 1]
After:
[3, 1, 1, 1, 1]
[5, 2, 2, 1, 5]
[7, 3, 2, 7, 6]
[8, 9, 6, 8, 7]
```
