Assignment 1

Julien Corman, Werner Nutt
Flavio Vella

Array Utility Class, Euclidean Algorithm

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.

1. Implementation of the basic operations of the class ArrayUtility

Implement in Java the class `ArrayUtility`, which offers basic operations over one-dimensional and two-dimensional arrays. *All* methods *must* be implemented as class methods (i.e., `static` methods). The signature of the methods in the `ArrayUtility` class are the following:

1. `public static int findMax(int[] A, int i, int j)`: returns the maximum value occurring in array `A` between position `i` (included) and `j` (excluded).

2. `public static int findMinPos(int[] A, int i, int j)`: returns the position of the minimal number v in array `A` between position `i` (included) and `j` (excluded). If v is repeated, returns the position of any of its occurrences.

3. `public static void swap(int[] A, int i, int j)`: swaps the elements in position `i` and `j` in array `A`.

4. `public static void shiftRight(int[] A, int i, int j)`: shifts to the right all elements of array `A`, starting from position `i` and until position `j` (i.e., moves the element in position k to position $k+1$ for all `i` $\leq k <$ `j`, and leaves position `i` unchanged).

5. `public static int[][] createRandomMatrix(int rows, int cols, int min, int max)`: creates and returns a two-dimensional array with `rows` rows and `cols` columns of random elements. Each element has a value between `min` (included) and `max` (included). Use the `Math.random()` method of Java.

6. `public static int findInArray(int[] A, int q)`: returns the position of number `q` in array `A`. Returns -1 if `q` is not present in `A`. If `q` is repeated, returns the position of any of its occurrences.

7. `public static int findInSortedArrary(int[] A, int q)`: behaves like `findInArray(int[] A, int q)`.

   The method assumes that the array `A` is *sorted* (in ascending order). It need not be correct if `A` is not sorted. Exploit the fact that the array is sorted to find an efficient algorithm (remember the Google interview questions!).

8. `public static int findFirstInSortedArrary(int[] A, int q)`: returns the position of the *first* occurrence of number `q` in array `A`. Returns -1 if `q` is not present in `A`.

   As before, the method assumes that `A` is sorted (in ascending order), and need not be correct if `A` is not sorted. Again, exploit the fact that the array is sorted to find an efficient algorithm.

For methods 7 and 8, write down an explanation (in English) of your solution to the problem.

(Weight: 60% of this CW)

2. Euclidean Algorithm

The algorithm `gcd(int a, int b)` is a simple version of the Euclidean Algorithm, which computes the greatest common divisor of two positive integers `a` and `b`:

```
int gcd(int a, int b) {
    while (a != b) {
        if (a > b)
            a := a - b;
        else
            b := b - a;
    }
    return a;
}
```

Write down proofs for the following statements:

1. The algorithm `gcd(int a, int b)` terminates for all positive integers `a` and `b`.
2. The algorithm `gcd(int a, int b)` is correct, i. e., always computes the greatest common divisor of `a` and `b`.

(Weight: 40% of this CW)

Deliverables. For Question 1, submit two copies of your code:

- one via Codeboard (instructions are available here),
- one via the OLE submission page of your lab (together with the other deliverables).

For Question 1 (methods 7 and 8), hand in a PDF containing your explanations.
For Question 2, hand in a PDF containing your proofs.

Combine all deliverables into one zip file, which you submit via the OLE submission page of your lab. Please include name, student ID and email address in your submission.

Submission until Thursday, 18 October 2018, 23:55, to Codeboard and the OLE submission page of:

Lab A / Lab B