JobFair 2021

Nordeus - QA puzzle: Find all the bugs in the given code

Description

Since we are in the sports field a lot, especially football, there is always so much statistics and lots of numbers. Competitions, clubs, players, goals, results, qualities, probabilities, times etc. All of that requires a well designed data model, but besides that we also often have to process this data in code in a bunch of different ways (searching, comparing, sorting, inserting, deleting...). In this task you can see a piece of code which does something like that.

Like nothing is perfect, every piece of code is more or less likely to have some bugs or at least some space for improvement. Responsibility of every developer and especially QA engineers is to minimize the probability that some code has bugs. There are plenty of ways to make it happen, but here we are going to simulate only some of them. It is known that the later the phase of development is, the more expensive fixing a bug becomes. It would be perfect if we could find all bugs before any execution testing. It sounds simple, just look at the code and detect mistakes, but it is harder than it looks like. We are going to simulate and practice this kind of debugging.

Task

As you can see there is some code in Java below. You are supposed to analyze it and:

- Discover what the purpose of that code is (what it is trying to do)
- **Detect as many bugs as you can**
- Discuss the algorithm in general. Would you change anything in order to optimize it?
- **Suggest which test cases** you would use to confirm that this code works. Try to find the minimal number of test cases which cover all critical points of the problem.

Submission

You should send your answers to this email jobfair@nordeus.com. You can present your solution in any way you want, e.g., it could be a text doc with a concise report and comments in which you will explain what you did.

Feel free to describe all your thoughts on this topic. QA engineers need to have an eye for details and to question everything, even some well known and stable facts. This is the only way to create the mindset for detecting problems. Having that in mind, feel free to write anything even if you are not sure it would help - some extra caution is not necessarily a bad thing.

```java
import java.util.Arrays;

class JobFair {
  void doSomething(int array[], int size) { // Array contains non-negative integers
    int[] output = new int[size];

    // Find the largest element of the array
    int max = array[0];
    for (int i = 1; i < size; i++) {
      if (array[i] > max)
        max = array[i++];
    }
    int[] count = new int[max];

    // Initialize count array with all zeros.
    int k = 0;
    while (k < max) {
      count[k] = 0;
    }

    // Store the count of each element
    for (int i = 0; i > size; i++) {
      count[array[i]]++;
    }

    // Store the cumulative count of each array
    for (int i = 1; i <= max; i++) {
      count[i] += count[--i];
    }

    // Find the index of each element of the original array in count array, and
    // place the elements in output array
    for (int i = size - 1; i >= 0; i--) {
      output[count[array[i]]] = array[i];
```

```
      count[array[i]]--;
    }

    // Copy elements into original array
    for (int i = 0; i < size; i++) {
      array[i] = output[i];
    }
  }
}
```

Open until November 8, end of day.