CSE 100 Algorithm Design and Analysis Lab 03 (20 points)
Due date: September 27, 23:59

# Radix-Sort

**Description** In this assignment you will implement RadixSort.
For testing your code you will use "Grade03", which should be in your working directory. Subdirectory "testfiles" contains files t1, t2, ..., t10 and o1, o2, ..., o10: for each testfile tx, the correct answer is in file ox. After you compile your code, your execution file's name must be "RadixSort.exe". Perhaps you may need to change the permission of "Grade03" as explained in previous labs.

If you run "./Grade03" and your code is correct, you will see the following messages:

```
test 1: success
test 2: success
test 3: success
test 4: success
test 5: success
test 6: success
test 7: success
test 8: success
test 9: success
test 10: success
```

You can also find a summary of the execution in the file named "result".

**Input structure** You are going to apply Radix-Sort to sort vectors. The input starts with an integer number which indicates the number of vectors to be sorted. Then vectors follow, one vector per line. Each vector consists of 10 digits where each digit has a value in $0 \dots 9$. Entries of a vector are separated by a space.

**Output structure** Output the sorted sequence of vectors, one per line. Vector $i$ must appear before vector $j$ in your output if and only if for some $d \in \{1, 2, \dots, 10\}$, vector $i$ is smaller than or equal to vector $j$ on the $d$th entry, and the two vectors are equal for any of the first $d - 1$ entries.

## Examples of input and output

*Input*

```
5
3 3 3 3 3 2 2 2 2 2
2 3 2 2 2 2 2 2 2 2
1 3 0 0 2 1 0 0 0 0
1 3 0 0 2 2 0 0 0 0
2 3 2 1 2 2 2 2 2 2
```

*Output*

```
1;3;0;0;2;1;0;0;0;0;
1;3;0;0;2;2;0;0;0;0;
2;3;2;1;2;2;2;2;2;2;
```

```
2;3;2;2;2;2;2;2;2;2;
3;3;3;3;3;2;2;2;2;2;
```

**Submission** Submit a "zip" or "tar.gz" archive of your program through the assignments page of CatCourses by the deadline. Use your UCMNetID as the file name. Be careful since CatCourse strictly enforces the assignment deadline. You may be asked to compile, run, and explain the code to the TA to prove that you understand what you wrote.

**Grading** We provide 10 test cases for you to try your implementation. Each of them is valued at 1 point if executed correctly. We will use 10 additional test cases to check that your implementation is general enough. These are not provided to you and are also valued at 1 point each.

**Important Note** We will use plagiarism software to detect cheating. Offenders will be subjected to the UCM Academic Honesty Policy which states: *if any violation of the UCM Academic Honesty Policy is suspected in a course, the instructor of record must fill out the Faculty Report for Academic Misconduct.*