Exercise 4
Duplicate Detection

- Deadline: **Monday, 26.01.15**
- The admission to the exam requires *all* exercises to be solved.
- The exercises should be solved in teams of two students.
- The datasets and supplemental material can be found at network drive S:
  `\\fs3\bbs\DPDC`
- The submission system can be found at:
  `https://www.dcl.hpi.uni-potsdam.de/submit/`
- To solve this exercise, please submit a zip file containing the following items:
  - **<algorithm_name>.jar**: An executable Duplicate Detection algorithm.
  - **<algorithm_name>.zip**: The algorithm's source code.
  - **<algorithm_name>_docu.pdf**: Short documentation of the algorithm.
  - **<algorithm_name>_pres.pptx/ppt/pdf**: Two slides presentation of the algorithm.
  - **results.txt**: The results file listing all discovered duplicates.

Task 1: Duplicate Detection - A discovery algorithm

Use Case: A duplicate is a pair of two records that both represent the same real-world entity. Usually, duplicates constitute quality issues, which is why they need to be detected and corrected. In this exercise, we inspect an address dataset containing the names, streets, towns, and phone numbers of one million (fictional) persons. Our goal is to clean the address dataset from duplicate entries describing same persons.

Task: Write an algorithm that detects possibly many duplicates in a given dataset. At the same time, the algorithm should report possibly few false duplicates, i.e. record pairs that describe different real-world entities, because cleaning such records leads to data loss. The rules for your implementation are as follows:

a) Write your own algorithm and do not use existing duplicate detection frameworks. You may, however, use external libraries, e.g. for Levenshtein or Jaro-Winkler String comparisons, but we recommend to implement these on your own as well for exercise.

b) Your algorithm should take the path to the input dataset as command line parameter. Hence, the following command should execute your algorithm on the *addresses.csv*-file if this is located next to the jar-file:
```
java -jar <algorithm_name>.jar addresses.csv
```

c) The output of your algorithm is a file called *results.txt* that contains all discovered duplicates. Each line in this file is a comma-separated pair of two record IDs representing one duplicate each. If, for instance, the records 128 and 329, 245 and 453, and 353 and 972 are duplicates, the result file would look like this:
```
128,329
245,453
353,972
```
The ID of a record is the record's first value. Therefore, the following record has the ID 128:
```
"128","Herr","Jon","Pütz","Weserdeich","2","26919","Brake","60909"
```

Evaluation: In this exercise, we evaluate the precision and recall of the duplicates that you discover on the address dataset. It is, therefore, important to find a good similarity function for the record comparisons. To test your algorithm, you can use the cd and restaurant datasets. Both datasets can be found on the bbs share with their gold standards.

Hints: We do not evaluate the runtime of your algorithm, but you still need a good pair selection technique, because you do not have enough time to compare all records. To maximize precision and recall, you can also use clustering, parallelization, machine learning, or any other technique that supports your duplicate detection process. It could further help to calculate the transitive closure on your results, but note that most duplicates in the address dataset are pairwise duplicates, i.e. the cluster size is mostly two records.

Task 2: Documentation

Write a short (max one A4 page) documentation for your algorithm describing the algorithm that you implemented and the techniques that you used.

In the same document, answer the following questions:

a) How many duplicates did your algorithm find in the address dataset?

b) How long did the discovery take on the address dataset and what machine did you use?

c) Did you discover any challenges of your approach, e.g. in runtime or memory consumption?

d) Analyze the results that your algorithm has found: List 3 record pairs (the complete tuples) that you think are *true* duplicates and 3 record pairs that you think are *false* duplicates.

Task 3: Presentation

Prepare two slides for a short, 5 min presentation of your algorithm in the lecture. One slide about the algorithm and one slide about the duplicates you discovered.

Note that each team will present its work once!