The test focuses on tokenizing the text files provided. For this test, we are interested in identifying words and terms in the text, the text is all lower case and only contains single spaces and newlines.

Task 1: Identifying common words between documents

For this task, you need to generate a matrix, where each entry contains the number of unique common tokens (words) between each pair of documents. The output should include no headers for rows or columns. The matrix should be symmetric, and follow the numbering conventions of the files. The diagonal entries would be the count of unique terms in each document.

For example, each file name corresponds to its row and column, so 001,001 = the unique number of terms between file 001.txt and file 001.txt, 001,002 is the unique number of terms between file 001.txt and 002.txt, 002,003 is the unique terms between 002.txt and 003.txt and so on.

files	001	002	003
001	100	0	0
002	20	50	0
003	60	35	85

Where the file output will look like

```
100 0 0
20 50 0
60 35 85
```

each column delimited by a space, each row is a new line in the file,

- the output must be written to **results.txt** in the current working directory.
- the solution must take the directory where the data files are as an argument. i.e. java -jar solution.jar /home/blah/testdata
- the solutions must be in a runnable format when you submit it. (jar, python script, etc)
- must submit the full code and the results you produced.

Task 2: Identifying words by frequency

A bigram is a sequence of two consecutive tokens (or words).

For example:

the quick brown fox

jumps over the lazy dog

---the bigrams would be: the quick, quick brown, brown fox, fox jumps, jumps over, over the, the lazy, lazy dog

Across the entire body of text (or corpus) find:

1. the top 50 most frequent unigrams (single tokens or words),
2. the top 50 most frequent bigrams.

The output should

- A file named **results2.txt** in the current working direction
- first 50 lines should be top most frequent unigrams, that is 1 token per line
- next 50 lines should be top most frequent bigrams, that is 1 bigram per line, the bigram should be separated by space
- optional new line at the end of file
- the solution must take the directory where the data files are as an argument. i.e. java -jar solution.jar /home/blah/testdata
- the solutions must be in a runnable format when you submit it. (jar, python script, etc)
- must submit the full code and the results you produced.

Please feel free to ask questions for clarification.