Information Retrieval

Practical session n°1: Inverted index and boolean retrieval

Team project "Information Retrieval":

- Team: 4 students.
- Team name: concatenation of the 4 first names, without spaces or special characters (e.g.: VictorAlbertJulesIsaac).

Create a directory named *practice1*. In this directory, create a new file named *practice1_report.txt*.
During the practical session, for each exercise, copy-paste in this file some outputs of your program, showing that you complete the exercise and it works correctly. Add some explanations.
At the end of the practical session, copy-paste the source code of your program(s) in the directory *practice1*.
Compress the directory in a file named *practice1_YourTeamName.zip* (or *.tar, .gz, .rar,* etc.) (e.g.: *practice1_VictorAlbertJulesIsaac.zip*).
Upload this compressed file (one file / team) on the website of the course **(deadline September 12th)**.

Exercise 1: Inverted Index

This is a collection of ten documents (one line, one document). A document is introduced on a new line by the sequence: `<doc><docno>document identifier </docno>`
A document ends with the string `</doc>`.

```
<doc><docno>D0</docno>Citizen Kane</doc>
<doc><docno>D1</docno>Casablanca</doc>
<doc><docno>D2</docno>The Godfather The Godfather</doc>
<doc><docno>D3</docno>Gone with the Wind</doc>
<doc><docno>D4</docno>Lawrence of Arabia</doc>
<doc><docno>D5</docno>The Wizard of Oz The Wizard of Oz</doc>
<doc><docno>D6</docno>The Graduate</doc>
<doc><docno>D7</docno>On the Waterfront</doc>
<doc><docno>D8</docno>Schindler's List</doc>
<doc><docno>D9</docno>Singin' in the Rain</doc>
```

Build (manually, in text or spreadsheet files) its inverted index usable for a boolean search. Give a represention of this index both with postings lists and with an incidence matrix.

Exercise 2: Toy indexer

Write a program in the language of your choice (recommended: perl, java, python, etc.) which index documents of the collection of Exercise 1 contained in a single file.

The first character of the document immediately follows the '>' of `</docno>`.
The content of the document only contains printable characters (letters, digits, punctuation signs, spaces, new lines, tab characters) except '<' and '>'. Every characters belong to the ASCII character set.

The index will be built and printed. It must allow an efficient access by term, and for each term it must contain the list of documents which contain this term. The length of this list is called the document frequency (`df`) of this term.

In a second version of the program, the number of occurrences of each term in each document (the term frequency (`tf`)) will be stored.

Carefully choose of data structures for the dictionary and the postings lists, being aware the impact of this choice on algorithmic complexity.

Here is the expected output (except for formatting details):

```
1=df(arabia)
      1 D4
1=df(casablanca)
      1 D1
1=df(citizen)
      1 D0
1=df(godfather)
      2 D2
1=df(gone)
      1 D3
1=df(graduate)
      1 D6
1=df(in)
      1 D9
1=df(kane)
      1 D0
1=df(lawrence)
      1 D4
1=df(list)
      1 D8
2=df(of)
      1 D4
      2 D5
1=df(on)
      1 D7
1=df(oz)
      2 D5
1=df(rain)
      1 D9
1=df(s)
      1 D8
1=df(schindler)
      1 D8
1=df(singin)
      1 D9
6=df(the)
      2 D2
      1 D3
      2 D5
      1 D6
      1 D7
      1 D9
1=df(waterfront)
      1 D7
1=df(wind)
      1 D3
1=df(with)
      1 D3
1=df(wizard)
      2 D5
```

Exercise 3: Boolean Queries

Implement the operators AND, OR and NOT (or AND NOT) of postings lists.
Build a boolean query parser.
Given a boolean user query, your program should be able to print a list of documents.
Try several queries in order to test the operators and their combination.