Week 3:
Index Creation

- This assignment is due on **4th November, 2015 (13:30)**
- You can discuss the problems with other groups of this course or browse the Internet to get help. However, copy and paste is cheating.
- There are 13 weekly exercises in total. In each one of them, all assignments sum up to 20 points. You need to achieve at least 80% of all assignments during the course in order to participate in the final exam. Hence, you need to achieve at least 208 points in total (*13*20*0.8=208*).
- Submission at
  `https://www.dcl.hpi.uni-potsdam.de/submit`
  - only pdf files
  - one file per group per week (week3.pdf)
  - put your names on *each* sheet in the pdf file

Assignment 1: Dictionary Compression

Given a document collection of 550,000 terms, let us assume that we sort the terms and store them sequentially in a large string (S). For instance, a part of S could be "`access,application,book..`". In order to identify where each term begins in S, we need to store their positions in the string. Thus, we use an array (A) of fixed-width entries to store information about the terms. The rows of A correspond to the terms and the columns are defined as follows: (*Freq*): the document frequency of a term t, (*Post*): the pointer to its posting list, (*Dict*): the pointer to its position in S.

a) After tokenizing the terms, the dictionary consists of 400,000 tokens. Assume that for each token we need 4 bytes for storing *Freq*, 4 bytes for *Post* and 3 bytes for *Dict*. We also use 8 bytes to represent each token in S. Which is the average number of bytes per token that we need in total? Which is the size of A? **3 P**

b) Additionally to the above setting, let us improve the dictionary space by using *blocking* – dividing S into groups (blocks) of k tokens. Consider also that k is equal to 4. We need to use one pointer of 3 bytes per group (to point at the first word of the group), which saves us some space eventually.

- It is important to distinguish the tokens that each group contains. Thus, we will use their length to seperate them in S. For instance, a block in S should look like : "`6medium6memory7message6method`". In addition, we need 1 extra byte to store the length of each token.
- How many bytes are we saving in total for each group of 4 tokens by using *blocking*? **5 P**

Assignment 2: Posting List Compression

Encode the following posting lists. A posting list comprises pairs of (document ID, term frequency)

a) (2,1), (4,2), (5,2) → delta encoding **3 P**

b) (2,1), (4,2), (5,2) → unary encoding **3 P**

c) (6,3), (7,1), (10,4) → Elias gamma encoding **3 P**

Assignment 3: (Programming) Index compression

This week we will build a compressed version of the index generated last week.

- We want to construct a compressed index structure, in a sense of compressing both the dictionary of the patent data and the posting lists. You can choose the method that you prefer for each task. The goal is to reduce the memory consumption so that a larger part of the index could fit into the main memory. Hence, response time will be also improved.
- You need to override the following functions `abstract void compressIndex(String directory)` and `abstract void loadCompressedIndex(String directory)`. In the first one you will build your compressed index and in the second one you will load its seek list to the main memory, so that you start answering queries. You will override them in your implementation file, namely *SearchEngineMyTeamName.java*. Note that the seek list may not fit into the main memory when we scale up the implementation and start using the overall patent dataset.
- Although the index will now be compressed, you should still use a seek list, so that you can access it faster. In addition, it is probable that the seek list needs to be adjusted to the new version of the index that you will build in this assignment.

a) Print the list of patent invention titles matching the following queries.
   - ”file-system” **1 P**
   - ”included” **1 P**
   - ”storing” **1 P**

b) Report the sizes of the compressed and uncompressed index files in your implementation.

c) As in the previous assignment, write down which kind of compression techniques you are using.