Data compression (up to 5 points) If you decide to do this problem, send the answers by email to `niwinski@mimuw.edu.pl and hugo@mimuw.edu.pl`, together with your name or names, as well as the programming language and the operating system that you used. Attach a zip file to your mail, which should contain for each question you answered, the source code of your program, a script file that compiles this program and makes a demo, a text file that give comments and/or answers the question. Space and time efficiency of your programs will not be evaluated. Most important is the number of questions solved, and your ability to interpret the experimental results in terms of entropy, expected-length, and code optimality. Do not hesitate to add your own experiments or programs, and to give much comments.

1. For each language L in the set {Polish, English, French}, choose two texts in this language, of at least 50 pages [1] , and name them **L1.txt** and **L2.txt**.

2. Write a program **letterfrequencies** that reads a text file and outputs the set of letters used and their frequencies, and the associated binary entropy. Syntax should be something like **letterfrequencies polish.txt**. Display should be something like **A .134 b .126 ...**. Do not pay attention to possibly weird display on screen of accentuated letters. Compare results for the various texts you chose.

3. Write a program **shannon** that reads a text file and outputs the associated binary Shannon-Fano code. Syntax should be something like **shannon polish.txt**. Display something like **A 001 b 0001010 L 01 ...**. Write a program **huffman** that reads a text file and outputs the associated binary Huffman code.

4. Write a program **compress** that reads a text file **file.txt** and converts it to two files **file.shannon.txt** and **file.huffman.txt**. This two files should be sequences of **0** and **1** characters[2], and be equal to the coding of **file.txt** using the Shannon-Fano or Huffman codes computed by programs **shannon** and **huffman**.

For each text you chose and each code, compare the number of bits of the original text, the number of **0** and **1** of the encoded text, and the entropy of frequency distribution of letters in the original text. Considering two texts in the same language, is it efficient to compute a code from one of them and encode the other with this code? Discuss practical interest of this remark.

5. Write a program **blockcompress** that reads a text and compresses it using block encoding and shannon-fano and huffman codes. Block-encoding means that instead of coding couples of bytes, 4, 8 or more bytes are encoded at a time. Compare efficiency of block encoding and letter encoding: compare space efficiency, in terms of size of compressed file plus space needed to store the code, and also time-efficiency.

6 (additional 2 points). Write a program **wordcompress** that compresses a text using word encoding. Word encoding means that the input alphabet of

[1] I recommend strongly to use texts encoded in UTF-16. This way, almost any letter will be coded on two bytes. Only very rare characters use four bytes, and they should not appear in natural language texts. This way, programs should be simpler to write. On **gutenberg.org**, a lot of texts are available for free as well as a tool to recode them to UTF-16. See also **http://polish.joelonsoftware.com**, instructive page about character coding systems.

[2] You can also do "real compression" and encode them as sequence of bits but it is trickier.

the code is the set of words union the set of punctuation symbols. Compare efficiency with block compression. Optimize word compression in the following way. For couple of words u, v such that very often, word v is the immediate successor of u in the text, add **"u v"** to the input alphabet. For example, in an english text, **"it is"** will be coded as a unique word. Compare efficiency with word coding, and generate random texts.