# Second programming exercise: (De)Coding

**The compression techniques GIF and JPEG are becoming more and more expensive. Because of this, we will create our own compression technique. For the second programming exercise we will make a program that can code and decode a file!**
**The user is asked for the names of the original and target file. The compression rate of the document has to be shown on screen after running compression (this is the ratio between the number of read and written characters, there is some variation in this depending on the operating system you use).**
**The coding method will be as follows: for every sequence of k-characters that are the same (with k being larger or equal to two) we replace it with that character followed immediately by 'k' the number of times it is repeated. Characters that are by themselves do not change, nor should return / endline statements. For example the sentence "zooologists assist stewardesses..." will be coed as "zo3logists 10as2ist stewardes2es".**
**In order to be able to decode our messages later on we have to somehow take into account how we parse numbers. For example, e234 could mean e234 or ee34! To solve this problem we use backslashes in front of cases when an actual number is meant. This means that ABC11123ddd\efG\\\1 should be coded as ABC\13\2\3d3\\efG\\3\1 (note that backslash itself is also coded as \\). The official name for this type of coding is run-length encoding.**
**To give you some inspiration, and examples to test with I provided two examples. You can find these on the drive.**

- **Simple input: test1in.txt; fully coded: test2uit.txt— compression up to 69%.**
- **Difficult input: test2in.txt; fully coded: test2uit.txt— compression up to 72%.**

**Ask the user a few simple questions for the input and output file names. Read the input file only once, and write the contents to the output file only once.**
***Every symbol in the input file can only be read ONCE, and your code should only have ONE ifstream.get statament in it! Using more only makes things difficult for yourself, so should be avoided.***

# Remarks

- **We assume that the user is kind enough not to make any mistakes with the input. The input files should only be read once. We also assume that characters never occur more frequently than 10000 times in a row.**
- **Use the correct line endings. Every line ending in a file consists of a \n in UNIX and a Line Feed \r\n in Windows. Usually this is done automatically for you by the OS. We assume that the EndOfFile symbol is a line ending as well.**
- **Arrays may only be used for the filenames (or a string if you please). For reading and**

**processing the text you only need a few chars. The only header files that you should include areiostream and fstream (And string if you want to, use c_str ;) ). From a file you can only read the input with infile.get(...). Within the main-loop of the program you should only have one, and only one .get statement. It is also disallowed to put characters back into their original file.**

- **Remember that you should output an information block at the start of your program. Make good use of functions; make for example an information block function, functions to read input, a coding and a decoding function. Global variables are STRICTLY forbidden.**
- **Rough estimate of the program size: 200 lines**