# COMP9319 2017s1 Assignment 1: Run-length encoding

Your task in this assignment is to create a simple compression program that implements run-length encoding (RLE). Your program will produce a file of the minimum size using only run-length encoding. Hints on representing the runs to achieve a minimum size will be discussed in the lecture. The original input file may include any visible ASCII characters up to ASCII value 126. You can use the first bit of a byte to determine if a byte represents an ASCII character, or if it is part of a number that represents the count.

Your C/C++ programs, called **rlencode** and **rldecode**, will accept either one or two arguments.

## rlencode and rldecode: Two arguments

When rlencode and rldecode accept two arguments, the first one is the path to an input file; and the second argument is the path to an output file. For example, for rlencode, the input file is the original input text file and the output file will be the RLE encoded file.

As described above, a visible ASCII character is 7-bit code, with the first bit of the byte being zero. To differentiate a count from a character, a count will have the first bit on, and use the remaining 7 bits to represent the actual value. When a number is larger than 2^7, you will need to concatenate the subsequent "count" byte(s) to form the final value. You are responsible to design the representation of these counts so that you achieve the minimum encoded file size.

## rlencode and rldecode: One argument

When rlencode and rldecode are given with one argument, which is the input file, they will output the RLE encoded representation of the input file to the standard output for debugging purpose (with each count printed as an integer enclosed by square brackets).

For example, suppose that the content of the input ASCII file called simple.txt is

```
aAAbbbBBBBccccccCCCCCCdDDeeeEEEE
```

Then

```
> ./rlencode simple.txt
aAAb[0]B[1]c[2]C[3]dDDe[0]E[1]

> ./rlencode simple.txt simple.rle
> ./rldecode simple.rle
aAAb[0]B[1]c[2]C[3]dDDe[0]E[1]
```

Note that from the example, we observe that rlencode assumes a normal ASCII text file as input and rldecode assumes an RLE-encoded file as input, even though both of them produce the same output. Please also note that to minimize the size of a count, I represent 3 runs as 0 (where 3 is the minimum number used to represent runs using count).

Your solution is not allowed to write out to any external file that other than the output file specified in the

command argument.

We will use the `make` command below to compile your solution. Please provide a makefile and ensure that the code you submit can be compiled. Solutions that have compilation errors will receive zero points for the entire assignment.

```
make
```

Make sure you put both "rlencode" and "rldecode" as your make targets. Your solution will be compiled and run on a typical CSE Linux machine e.g. wagner. Your solution should **not** write out any external files other than the index file. Any solution that writes out external files other than the index file will receive zero points for the entire assignment.

# Performance

Your soluton will not be tested against any ASCII text files that are larger than 200MB.

In the testing machine, runtime memory is assumed to be always less than 20MB. Runtime memory consumption will be measured by `valgrind massif` with the option `--pages-as-heap=yes`, i.e., all the memory used by your program will be measured. Any solution that violates this memory requirement will receive zero points for that query test. Any solution that runs for more than **60 seconds** on a machine with similar specification as *wagner* on a given text file will be killed, and will receive zero points for the tests for that file. We will use the `time` command and count both the user and system time as runtime measurement.

# Documentation

You will be marked on your documentation of your comments for variables, functions and steps in your solution. Your source code will be inspected and marked based on readability and ease of understanding.

# Assumptions/clarifications/hints

1. The input filename is a path to the input file. Please open the file as read-only in case you do not have write permissions.
2. Marks will be deducted for output of any extra text, other than the required, correct answers (in the right order). This extra text includes (but not limited to) debugging messages, line numbers and so on.
3. Your implementation of RLE must be lossless, i.e., after running rldecode on the encoded file generated by rlencode, we should get back a file exactly the same as the original ASCII text file.
4. Marks will be deducted if your RLE encoded file is not minimum in size, due to non-optimal representation of the counts.

# Marking

This assignment is worth 100 points. Below is an indicative marking scheme:

| Component | Points |
|---|---|
| Auto marking | 90 |

| | |
|---|---|
| Documentation | 10 |

# Bonus

Bonus marks (up to 5 points) will be awarded for the solution that achieves 100 points and runs the fastest overall (i.e., the shortest total time to finish **all** the tests). This solution will be shared with the class. Note: regardless of the bonus marks you receive in this assignment, the maximum final mark for the subject is capped at 100.

# Submission

**Deadline: Sunday 26th March 23:59**. Late submission will attract 10% penalty for the first day and 30% penalty for each subsequent day. Use the give command below to submit the assignment:

```
give cs9319 a1 makefile *.c *.cpp *.h
```

Note: **Please use classrun to check your submission to make sure that you have submitted all the necessary files.**

# Plagiarism

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this subject. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.