COMP9319 2020T2 Assignment 1: Arithmetic Coding

Your task in this assignment is to implement the static Arithmetic Coding algorithm (both encoder and decoder) as presented in the lecture week 1 in C/C++. You will also need to provide a makefile to compile your source files and generate an executable program running on a typical CSE Linux machine. In case if you have never created a makefile before, a sample C program with a very simple makefile is available here. You shall modify it accordingly for your assignment solution (e.g., g++ instead of gcc; different c, cpp or header files; include different libraries). You may include and use any C or C++ libraries available in CSE linux machines.

Your encoder and decoder (called `aencode` and `adecode`, respectively) will take input from the standard input, and produce output to the standard output.

The encoder should output the AC encoded value in decimal digits (for learning purposes), instead of conventional binary output. Similarly, the decoder will read the encoded message in decimal digits as input. Please note that your program needs to support computations to the required precision (which may be more than the maximum size of the default floating numbers of C/C++). You may use any libraries available in CSE linux machines to achieve this, e.g., refer to the provided sample C program above.

The input to the encoder is the source content to be encoded. It can be input via the terminal and terminated by a Ctrl-D ("^D"). The output of the encoder will contain: the frequency of each symbol (sorted lexicographically in ascending order and one line each); the low and the high (separated by a space) of the final result at the last line. We assume that the symbols will divide the number line [0, 1) according to their lexicographical order (i.e., similar to the "BILL GATES" example discussed in the lecture). The source content may contain any visible ASCII symbols (e.g., alphabets, punctuations) and spaces. Your program will not be tested with content containing newlines, invisible symbols, or tabs.

The input to the decoder is the same as the output from the encoder, except that the last line contains a decimal number (the AC encoded value of the source content), plus optionally a space and then some comment text. Your decoder shall ignore the text after the first decimal number (the AC encoded value) of the last line. The output of the decoder shall output the content identically to the original source content before the encoding.

Your assignment will not be tested with any input that is more than 1024 characters. Marks will be deducted if your solution outputs any extra text, other than the required, correct output as shown in the examples below. This extra information includes (but not limited to) debugging messages, line numbers and so on. 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. Any solution that fails to compile on a CSE Linux machine, or writes out external files, will receive **zero** points for the entire assignment.

Your submitted file(s) will be compiled using the following command on a CSE linux machine:

```
%wagner> make
```

This will generate two executables, namely `aencode` and `adecode`.

Any solution that fails to compile on a CSE Linux machine with the above command will receive **zero** points for the entire assignment.

Usage and Example

```
%wagner> cat a.txt
BILL GATES%wagner>
%wagner> aencode < a.txt
  1
A 1
B 1
E 1
G 1
I 1
L 2
S 1
T 1
0.2572167752 0.2572167756
%wagner>
%wagner> cat a.txt | aencode | adecode > b.txt
%wagner> diff a.txt b.txt
%wagner>
```

Since adecode only read the first decimal number of the last line and ignore the rest of the line, we may modify the output above and use any number in the interval of [0.2572167752, 0.2572167756), then the decoder will still generate the same output. For example:

```
%wagner> cat c.txt
  1
A 1
B 1
E 1
G 1
I 1
L 2
S 1
T 1
0.2572167755 # This is a legit AC encoded value
%wagner>
%wagner> cat c.txt | adecode > d.txt
%wagner> diff a.txt d.txt
%wagner>
```

Compile

We will use the commands below to compile your solution. Please ensure that the code you submit can be compiled. Any solution that has compilation errors will recieve zero points for the entire assignment.

```
make
```

Your solution will be compiled and run on a typical CSE machine e.g. wagner.

Documentation and Code Readability

We assume a reasonable quality of your code readability and documentation in your submitted code. Marks may be deducted if your code is difficult to read and/or understand.

Submission and Marking

Deadline: Monday 29th June 09:00AM. Late submissions will attract a 1% penalty of the total mark achievable for this assignment per hour after the deadline (i.e., 24% per day), and no submissions will be accepted after 3 days late. Use the give command below to submit the assignment:

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

Or submit via WebCMS: Login to Course Web Site > Assignments > Assignment 1 > Assignment 1 Specification > Make Submission > upload required files > [Submit]

This assignment is worth 15 points, and will be auto-marked. Your code will also be checked manually for readability and plagiarism.

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. Please refer to Student Conduct/Plagiarism section of the Course Outline (at WebCMS) for details.