CSC 485B/578B - SUMMER 2023
DATA COMPRESSION
ASSIGNMENT 1
UNIVERSITY OF VICTORIA

Due: Sunday, May 21st, 2023 at 11:59pm. **Late assignments will not be accepted.**

This assignment will be submitted electronically through Brightspace (as described in 'Submission Instructions' below). All code submissions must be your own work. Sending code to other students, receiving code from other students, or exchanging assignment code with anyone else via any medium is plagiarism and is prohibited.

1 Overview

Your task is to create your own implementation of the old Unix `compress` program in either C or C++. You will only implement the compressor; to decompress, you can use the decompressor for the real `compress` program. For example, once your program (called `uvcompress`) is complete and correct, you can compress and decompress files with the following commands.

```
$ ./uvcompress < my_input.txt > compressed.Z
$ compress -d < compressed.Z > decompressed.txt
```

(After these commands, if your compressor worked correctly, the `decompressed.txt` file should be byte-for-byte identical to the original input `my_input.txt`).

Your `uvcompress` program must implement the following requirements.

- The produced bitstream must be compatible with the `compress -d` command on our common reference server (`csc485b.csc.uvic.ca`).
- The width of symbol values starts at 9 bits and increases (as specified in the 'Unix `compress`' lecture) to at most 16 bits, after which point the symbol table stops growing (but the program will continue to accept input and generate output).
- Input must be accepted from standard input (not from a file on disk).
- Output must be generated to standard output.
- The 'reset marker' feature must **not** be used anywhere in your generated bitstream. However, your program should still reserve entry 256 in the symbol table for the reset marker (to ensure compatibility with the `compress` program).

More details on the `compress` program and its underlying architecture are available in Lectures 3 and 4.

2 Submission Format

We will use automation to validate your submission before human inspection. To ease this process, we expect all submissions to be in a standard format.

Your submission must be a `.tar.bz2` archive called `uvcompress.tar.bz2` containing a directory called `uvcompress`. It must be possible to extract and run your submission using the **exact** following commands (starting in the directory containing the `uvcompress.tar.bz2` archive).

```
$ tar -jxf uvcompress.tar.bz2
$ cd uvcompress
$ make
$ ./uvcompress < /some/path/to/my_input.txt > compressed.Z
```

(where the input file is some local file not contained in your archive). To make the process easier, two starter archives have been provided (containing some placeholder code): `a1_starter_c.tar.bz2` contains a basic Makefile and source file for C code, and `a1_starter_cpp.tar.bz2` contains a basic Makefile and source file for C++ code.

To create an archive to submit, run the following command (in the parent directory of your `uvcompress` directory):

```
$ tar -jcvf uvcompress.tar.bz2 uvcompress/
```

(the `-j` flag is used to enable BZip2 compression of the output).

3 Test Data

An archive of test inputs has been provided, containing the Calgary corpus, the Canterbury corpus and some other miscellaneous inputs (mostly from lecture examples). The test data archive also contains the output of a model `uvcompress` implementation on each file (the compressed files use the customary `.Z` extension).

For small inputs, the compressed data from `uvcompress` should be identical to that produced by `compress`. On larger inputs, the compressed data may differ since the `compress` program may emit the reset marker (which you are not permitted to use). However, your output should be identical to the compressed model output in the test archive in all cases.

4 Evaluation

Your submission must be an archive named '`uvcompress.tar.bz2`' containing a directory `uvcompress` (which will contain a Makefile and your code, as described above). Your code must compile and run correctly on `csc485b.csc.uvic.ca`. If your code does not compile as submitted, you will receive a mark of zero.

This assignment is worth 5% of your final grade and will be marked out of 10 as follows.

Marks	Component
9	Your implementation generates correct output (identical to a model solution) on all of the provided input files and requires less than 30 seconds to compress the entire dataset.
1	Your implementation follows a streaming protocol, where the amount of input or output data stored in memory is fixed. The symbol table may grow as needed. This mark will only be given if at least 6/9 was achieved on the item above.

Submission Instructions

All submissions for this assignment will be accepted electronically. You are permitted to delete and resubmit your assignment as many times as you want before the due date, but no submissions will be accepted after the due date has passed.

Ensure that each file you submit contains a comment with your name and student number, and that the files for each question are named correctly (as described in the question). If you do not name your files correctly, or if you do not submit them electronically, it will not be possible to mark your submission and you will receive a mark of zero.

If you have problems with the submission process, send an email to the instructor **before** the due date.

It is your responsibility to ensure that we receive the correct file. To verify that you submitted the correct file, you can download your submission from Brightspace after submitting and test that it works correctly. **We will assume that all submissions have been tested this way**, and therefore that there is no reasonable chance that an incorrect file was submitted accidentally, so no exceptions will be made to the due date as a result of apparently incorrect versions being submitted.