CSC 212—Algorithms and Complexity— ***Huffman coding:*** (1) Compression and (2) decompression, and (3) an investigation on the effects of noise.

Practical 5 Term 3 12 August 2015

Write code that will read and compress the data in: `/export/home/notes/ds/ulysses.text`. There are three files that must be processed. First, for testing purposes process the file called `thecatshorter.text`, then test your code on the file `thecatlonger.text`, and, finally, run code on the large file called `text.txt`. Before running your code on the large file, ensure that it works by first running it *correctly* on the two `thecat<shorter|longer>` files. Do not copy these files into your own file system. Use then directly from the `notes` file. You may write your Huffman encrypted files to your own file system—but do not forget to delete them.

1. **The file system**: Put today's work in your `53practical` directory that contains a correct, appropriate working `Makefile`.
2. The program first needs to create Huffman code for a small string read from the file `thecatshorter.txt` and compress and decompress the file. Only when your code has been thoroughly tested for the small file should you proceed to the bigger files. There is very little new code since most of the code can be lifted from Sedgewick and Wayne's book, but you do need to add some code of your own. Most of your programming effort must go into measuring the effects of inserting noise or distorting the compressed file. What your experiment must determine how resilient Huffman code is to noise.
3. The nonzero frequency of each character together with its binary Huffman coding must be displayed.
4. The program must be capable of both compressing the data, and decompressing it by reading back the data from the stored Huffman-coded file.
5. Save your compressed file and then read it back to decompress it.
6. We repeat: *Do not copy the data*, read it directly from the files and use it to generate your results. Please do not leave the uncompressed input data files on your file system.
7. Fix your '`Makefile`' so that it deletes your encrypted files and decrypted file when running `make clean`.
8. **Effects of noise on Huffman coding** This is an experiment using the encrypted file data read back from the disk.
   (a) Read about 10 000 characters from the Huffman-coded file and disturb this input before decrypting it by introducing a little noise.
   (b) Introduce random noise into this file by flipping any 8 contiguous bits on a character boundary every 1000 or so characters starting at bit zero, before decrypting the input.
   (c) Your code must compare the pure uncoded input text with the text that is reconstructed from the damaged Huffman encoded file.
   (d) Determine the ratio of $\frac{\text{recovered code}}{\text{total–destroyed code}}$. In other words determine how much code can be recovered compared to what has not been destroyed.

(e) When you are satisfied that your code works for 8-bit errors every 1000 characters, you must do the same experiment by introducing a random number of contiguous flips, say $m = 8 * k$ where $k \in [1..8]$, every 1000 characters to try to make some conclusions about how long the decoding takes to recover automatically. Please use your discretion to alter the parameters if you see a way to improve this experiment.

(f) The experiment must be repeated 30 times and means and standard deviations must be calculated. Your code must collect and display these statistics and display the results.

9. The results must be properly written up using LaTeX in a report. An example file for such reports is available in the `.../notes/ds/report/` directory. Put the report in a directory parallel to your other pracs and call it `53huffmanreport`. This report must be submitted before or on 28 August 2015 by running a `make submit` with the report as one of your directories.

The `Makefile`s are all available in the `.../notes/ds/` directory in the Sunlab.
