This lab concerns how close a data set is to being sorted (in ascending order). Suppose that we have a data set of \( n \) items such that the list of values can be broken up into \( k \) contiguous in-order segments: \( S_0, S_1, \ldots, S_{k-1} \), each of length \( m_0, m_1, \ldots, m_{k-1} \), where \( \sum_{j=0}^{k-1} m_j = n \). For example, suppose that \( n = 8 \) and the list is:

\[ \{2.3, 3.5, 9.3, 4.2, 1.3, 2.8, 7.7, 7.9\} \]

so that \( k = 3 \) and the in-order segments are:

\[ S_0 = \{2.3, 3.5, 9.3\}, \quad S_1 = \{4.2\}, \quad S_2 = \{1.3, 2.8, 7.7, 7.9\} \]

with \( m_0 = 3, m_1 = 1 \), and \( m_2 = 4 \). We define the Shannon entropy \( E \) of the data set as follows. Let \( p_i = m_i/n \) for \( i = 0, 1, \ldots, k-1 \) and define

\[ E = -(p_0 \log_2 p_0 + p_1 \log_2 p_1 + \ldots + p_{k-1} \log_2 p_{k-1}) \]

So that if the list is already sorted in ascending order, \( k = 1, m_0 = n, p_0 = 1 \) and

\[ E = -(1 \cdot \log_2 1) = 0 \]

The worst case occurs when the data set is in reverse order and it is then clear that \( k = n, m_i = 1, p_i = 1/n \), and

\[ E = -n((1/n) \cdot \log_2(1/n)) = n((1/n) \cdot \log_2 n) = \log_2 n \]

Since entropy for a data set of \( n \) elements ranges from 0 to \( \log_2 n \), one commonly converts entropy into a percentage by dividing by \( \log_2 n \) and multiplying by 100.0%. If logarithms to base 2 are not available we can use common logs and the formula

\[ \log_2 x = \log x / \log 2.0 \]

1. Get the program \texttt{entropy.c} and compile and link it. Make several different data sets of around 20–30 numbers, some very nearly sorted and some very random, and try the program via

\texttt{entropy my_datafile.txt}

This program will also list your numbers if there are not too many of them. Try the program on some random datasets made with the \texttt{uniform.cpp} program. Do the entropy percentages agree with your perceptions of how close to sorted the data sets are?

2. Get the program \texttt{entropy_gen.c} and compile and link it. This program makes saw-tooth datasets, with some randomization, of a desired count and entropy (5 – 90%). Like \texttt{uniform.cpp} it writes the dataset to \texttt{stderr}. However, it may not be possible to make a small dataset with a very small entropy, and the program will warn you with an error message. In this case you should increase the size of the dataset. For example, to make a
sawtooth dataset of 100,000 numbers with a entropy of 25.0% you could type
`echo "100000 25.0" | entropy_gen 2> sawtooth_25.txt`
Test the sawtooth dataset you made with the `entropy.c` program to see that it is correct. Try other values of count and entropy. You should try enough values so that it is clear why we are attaching the name “sawtooth” to these datasets.

**Assignment** Use `entropy_gen.c` to make the following sawtooth datasets, each containing 100,000 numbers, with the following entropies (expressed as percentages): 5%, 10%, 15%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%.

Use the program `sorting.cpp` and time `quick_sort`, `heap_sort`, and `merge_sort` for each of these sawtooth datasets. Record your results in a text table – not in Excel. Write a short paragraph of your conclusions concerning which of these types of sorting algorithms is sensitive to entropy. Email me your results as in-line text. Please:

i. Keep line length under 80 characters per line.

ii. Do not send any attachments.