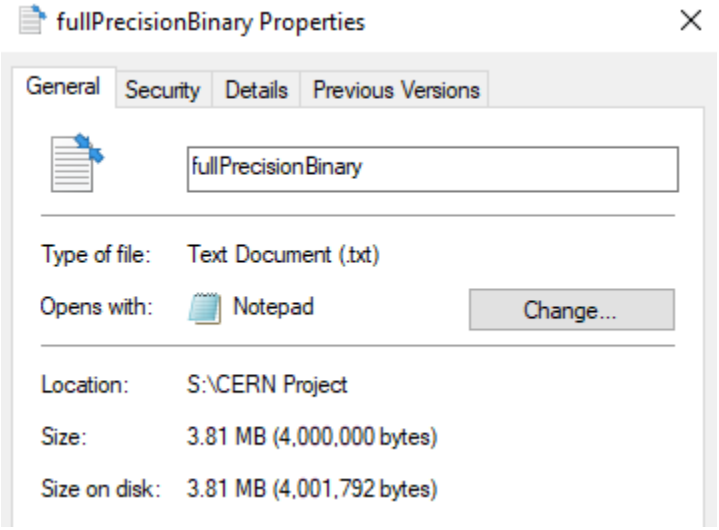
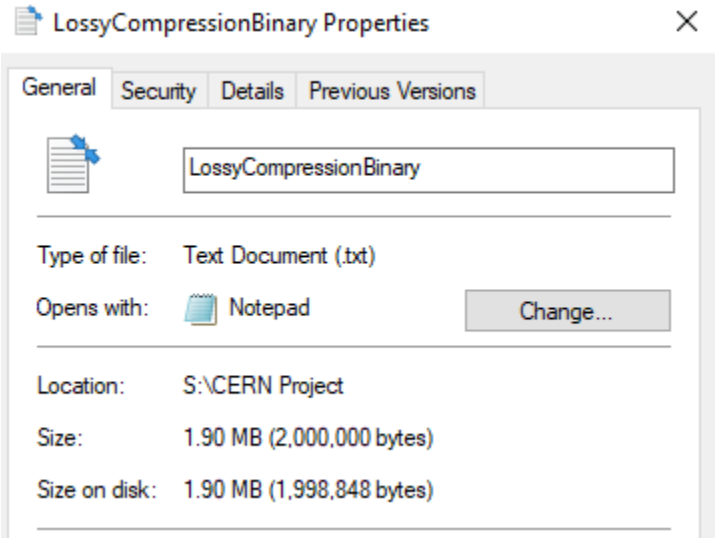
4) Measure and compare on-disk file sizes to analyze storage savings achieved through compression.

Comparing the on-disk sizes of the lossy compressed binary file and the non-compressed binary files shows us these results:



Size on disk (4,001,792 bytes)



Size on disk (1,998,848 bytes)

$$Percent\ Saved = \frac{1998848}{4001794} \times 100\% = 49.95$$

This means the amount of space saved after the lossy function is about 50%

5) Compare statistical parameters of the distributions before and after compression.

Using Excel to compare all 1 million plot points before and after shows us these results for various statistical parameters

	Before	After
Mean	49.97636	49.83318
Std Dev	28.87159	28.78821
Median	49.97555	49.75
Skew	0.001024	0.001229
Kurtosis	-1.20097	-1.20009

The mean has shifted down slightly due to losing some data points but it is still close to the predetermined point of 50 selected from the code

The standard deviation has decreased after the compression, implying that the spread of the data has been more smoothed and should be closer to the mean value

The median has also decreased which matches with the mean decreasing as well. This also shows that the values have slowly shifted downwards

The skew of the points is very close to zero, which means the compression did not change the distribution by a large amount. This is also a very small increase that can be negligible

Lastly, Kurtosis remained nearly unchanged, an indication that the tail behaviour of the distribution was not significantly affected by compression

6) Compute Mean Squared Error between original and compressed data.

Using the 1 million data points, the Mean Squared Error was found by following the formula below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

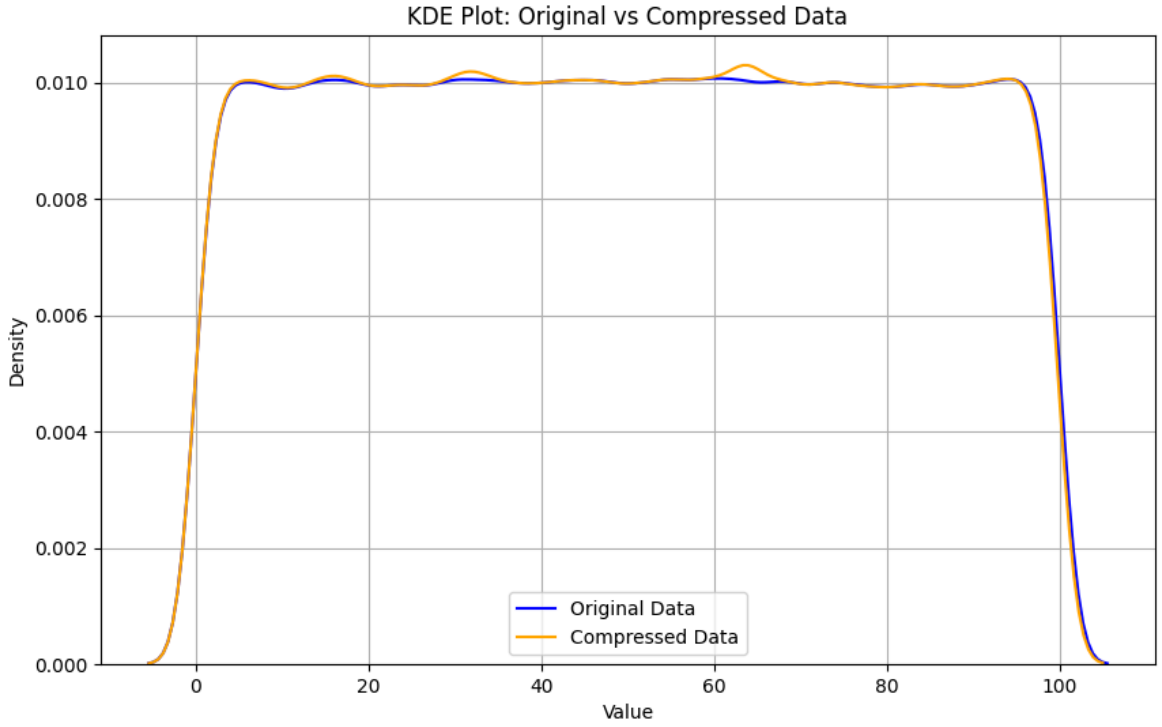
y_i is the non-compressed data and $\widehat{y}_i$ is the compressed data.
They were all subtracted from each other. Then they were squared and then all added together. Lastly, they were divided by 1 million. After doing all of that, the lossy function gave a Mean Squared error of **0.03753**. This is extremely small and shows that the compressed data is extremely similar to the original.

Using data points from a Gaussian Distribution set gives a Mean Squared error of
0.03476

Similarly using data points from an Exponential Distribution set gives a Mean Squared error of
0.03499

7) Plot original and compressed data to observe distribution changes.

Plotting the floating points from a uniform distribution gives a graph that look like this:



Looking at the graph shows that the distribution is very similar. The original data leans more towards the higher values whereas the compressed data can be seen shifting over to the lower values.

8) Discuss optimal compression levels for different use cases (e.g., high-precision computing or limited storage resources).

Overall the compression levels done by my program are by removing 16 bits from the mantissa. This causes the most level of inaccuracy due to losing the most bits. However even though it is the most inaccurate, it was still very accurate as shown from the graphs and calculations above. Since removing 2 bytes and being able to save with half as many and saving twice as much space, this should be used when there are limited storage resources. If once wanted to do high precision computing, one way to save space would be to remove only 1 byte or even half a byte from the mantissa. This would still save around ¼ to ⅛ the amount of space while still being very accurate.