Part 1: Compression (40 points)

old quote from Vangie Beal, managing editor of Webopedia…
(all lower-case letters have been used to simplify the example)

data compression is particularly useful in communications because it enables devices to transmit or store the same amount of data in fewer bits. there are a variety of data compression techniques, but only a few have been standardized. the ccitt has defined a standard data compression technique for transmitting and a compression standard for data communications through modems. in addition, there are file compression formats, such as arc and zip.

Compression algorithms substitute a repeating string in the original text with a unique token and keep a record of each substitution in a dictionary. The dictionary is stored with the compressed text for later decompression. (this is an analogy, not the math and science used by LZW or Huffman encoding)

♥*data*
♠*compression*
♦*communications*
☼*transmit*
♣*here are*
☯*technique*
☺*standard*

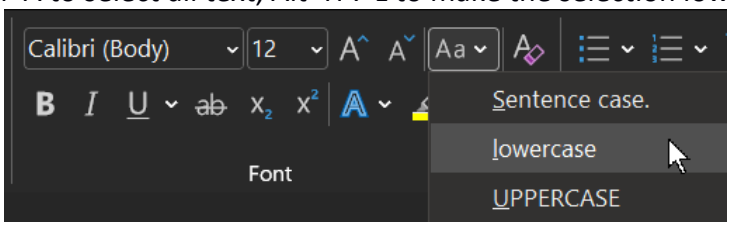
♥♠*is particularly useful in* ♦*because it enables devices to* ☼ *or store the same amount of* ♥*in fewer bits. t*♣*a variety of* ♥♠☯*s, but only a few have been* ☺*ized. the ccitt has defined a* ☺ ♥♠☯ *for* ☼*ting faxes and a* ♠☺ *for* ♥♦*through modems. in addition, t*♣ *file* ♠*formats, such as arc and zip.*

Including dictionary, total size is 368 characters or 82% of original.
The compression factor increases according to the frequency of repeating strings in the text.
Although Huffman and LZW compression routines are more sophisticated, the above illustrates the concept with character data instead of binary.

➔ How much can you compress the **lyrics to a song using the ideas above?**

You choose the song.

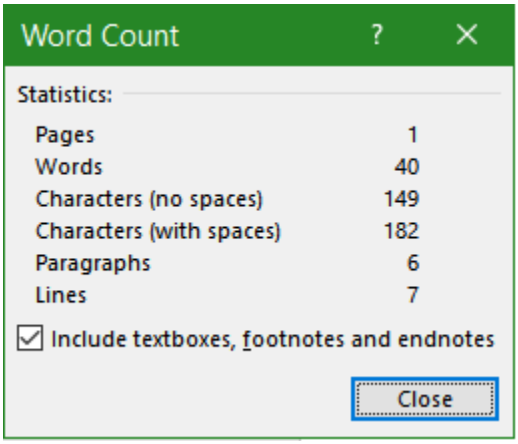
→ Copy the lyrics of a song to a new MS-Word document (Ctrl+N).
→ To reduce complexity, make all letters lower case:
Ctrl+A to select all text, Alt+H 7 L to make the selection lower case.



In the bottom left of the Word display, click "### words".
e.g. 

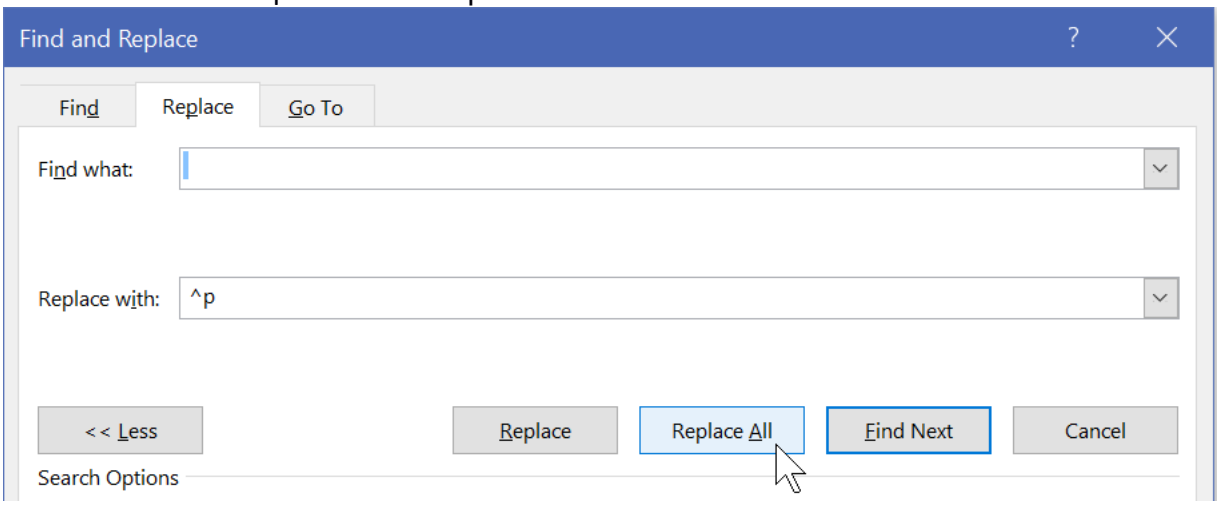
The Word Count dialog will pop up showing the number of characters with spaces.

N.B. paragraph / new line / CRLF characters are not counted by Word.
(Alt+H,8 will toggle the display of whitespace characters)

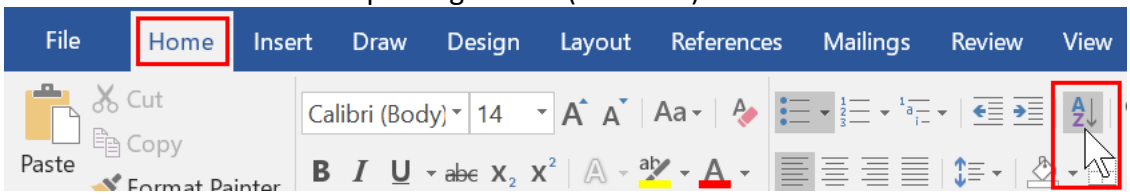


The following will help with your substitution analysis: separate the words in the text so each is on its own line, then sort the lines to see repeating patterns of individual words.

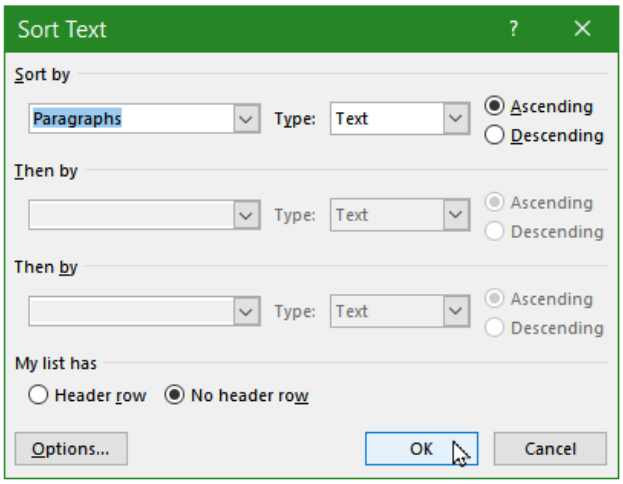
- copy the lyrics to another new document (Ctrl-N) used only for analysis
- Find and Replace a **space** with a **space + paragraph marker ^p** (Ctrl+H)
  Find what: □ Replace with: □^p



- Then sort the lines to see repeating words. (Alt H S O)



Sorting by Paragraphs results in one word per line, in alphabetical order; this makes it easy to see repeating words:



Anything occurring only once is not worth substituting with a token and including in the dictionary; you will be adding two characters (the tokens) to the file. Any string with a length of 2 or 3 and occurring only twice is similarly not worth it.

- a space is a *character* that can be compressed together with some word strings
- a robotic replacement of recurring words will not result in the best compression.
  - Consider whether a leading and/or trailing space should be compressed with a word
  - Consider repeating phrases before compressing individual words
- the token/string compression dictionary must be included to decompress back to its original.
  - The overhead of the dictionary must be included with the compressed text to assess the size of the compressed text versus the original.
- **Formula for characters saved:**
  - For the ***length*** of any string occurring ***n*** times and replaced by a single character token:
  - Saving ***n*** occurrences × ***length*** has the cost of a dictionary entry (token + string ***length***) plus ***n*** tokens replacing the string in the original text. For example,
  - `"Every "` occurs 3 times with a length of 6 including the trailing space (18 characters saved)
  - less the overhead of 7 characters in the dictionary, e.g. `"%Every ",` and three % tokens in the text taking the place of each `"Every "` string.
  - 18 saved less 7 added to dictionary less 3 replacements of text with a token = 10 characters of compressed text. 10/18 = 55.5% of the original.

➔ **How much can you compress the lyrics?**

- Use unique tokens – symbols that do not appear in the lyrics. E.g. the special characters and digits on the keyboard's top row.
  N.B. do not use the **^ carat** symbol, it is a Microsoft escape character which will confuse its Find & Replace process.
- Decompression reads the first character in the dictionary as the token and the next characters to end-of-line as the original string to replace tokens with.

Copy and paste the following into your answer document:

➔ The lyrics of your chosen song with attribution please.

➔ What was your dictionary of compression token to string characters, one entry per line?

e.g. (there is a space following the word because, to software, a space *is* a character)

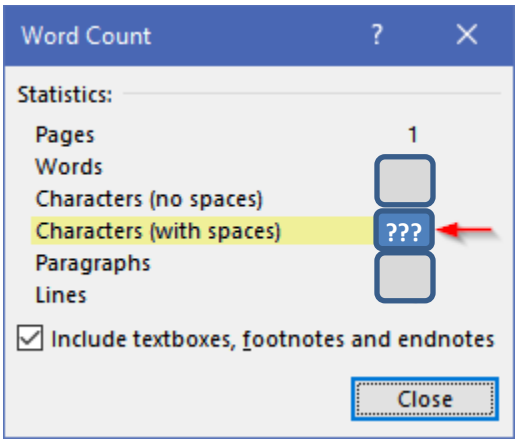
♥*data*
♠*compression*

➔ What were the compressed lyrics with the token substitutions?

i.e. the compressed text (not the sorted analysis list of words)

♥♠*is* …

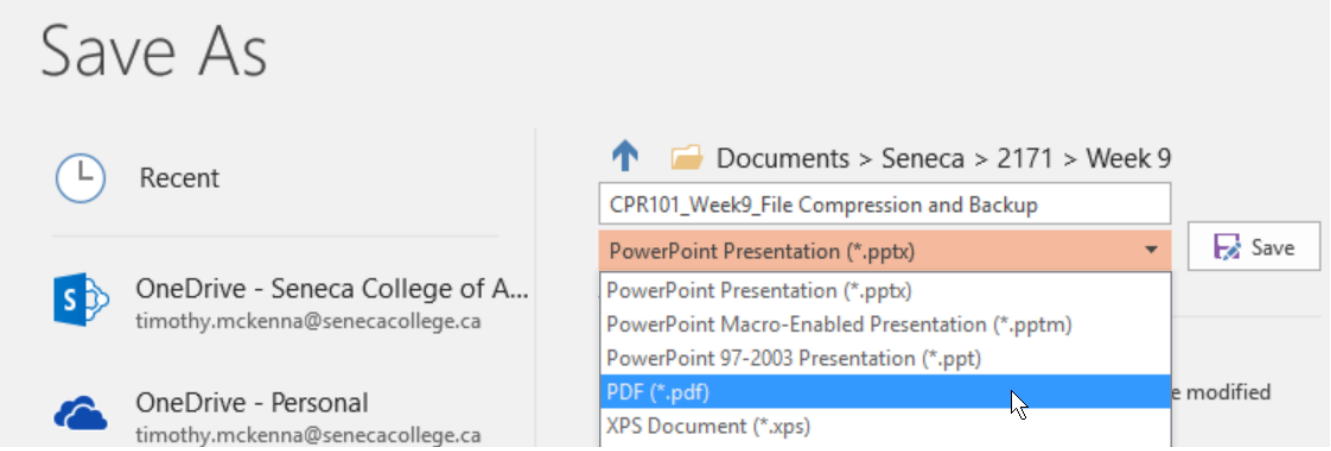
➔ What is total dictionary *plus* compressed text characters (with space) as a percentage of the original's size?



➔ **Test your compression dictionary by decompressing.** Process dictionary items from the bottom up: find the compression character in the compressed data and replace it with the original string. **Paste the decompressed version below** – *even if it is not perfect*. **What modifications, if any, does the compression dictionary need to return the compressed data back into its original state?**

Part 2:

- If it is not already there, extract / decompress the files in this week's activity .zip archive to your Desktop.
  - Remember that compressed files must be decompressed before they can be used. Windows does this automatically into the %temp% folder if you open a file directly from a .zip archive.
- Download this week's PowerPoint slides and save to your Desktop folder
  - open it, File menu, Save As, PDF (*.pdf)
  - File menu, Save As, PowerPoint 97-2003 Presentation (*.ppt)
    If you see the Microsoft PowerPoint Compatibility Checker, click Continue.



Compress all files from your Desktop folder into a zip archive:

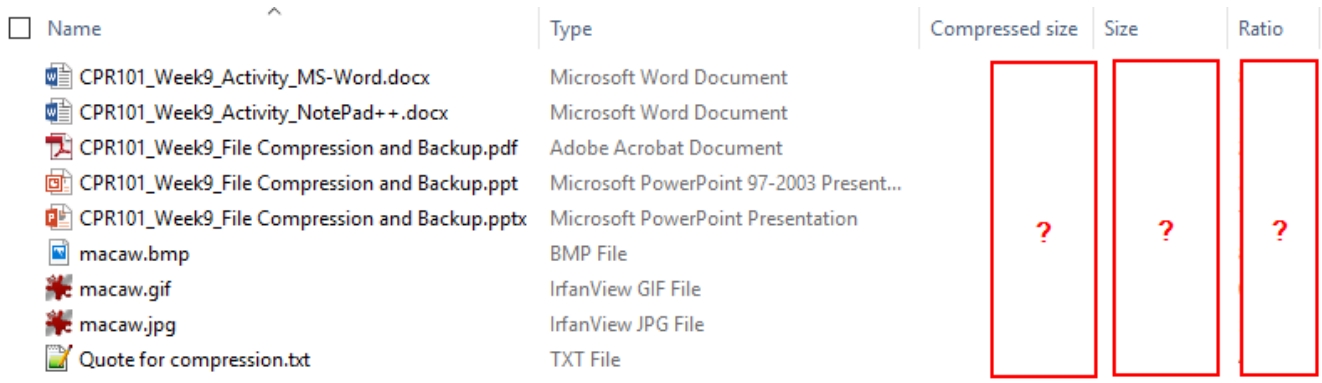
- select the files then right click and use the
  Send to > *Compressed (zipped) folder* option or use 7zip.

Open the .zip archive with Windows [File]Explorer.

On macOS, open a terminal window and cd to folder with .zip file

```
$ zipinfo -m archivename.zip
          -m [medium] shows percentage of file size saved by compression, higher is better.
          -l [large ] shows original and compressed file sizes in bytes.
```

Use the Snipping Tool or Snip & Sketch (⊞ + "snip") to copy only the information seen below.



The Ratio shows the proportion of space saved. `Ratio = (Size – Compressed) / Size * 100`

"Ratio" is a misnomer because it is not a ratio of the sizes shown. The "Ratio" of a calculated, non-displayed value relative to one of the two values shown is a guessing game, not a good user interface.

FYI: opening the .zip archive with 7zip will show bytes, not rounded K bytes, for original Size and Packed (compressed) size.

See https://www.noupe.com/design/everything-you-need-to-know-about-image-compression.html

➔ Paste the image of the Windows [File] Explorer .zip archive information or equivalent from macOS.

→ knowing the properties of different file formats is essential to answering the questions below.

Which image format should you use? See this.
Reduce the Size of Microsoft Office Documents using Word as an example

➔ Files with the **lowest** ratios were compressed the **least**. Ratio indicates % of space saved.
Which file types compressed the least? Why would that be? (10 pts)

➔ Files with the **highest** ratios were compressed the **most**.
Which file types compressed the most? Why would that be? (10 pts)

Part 3: Backup

The most common cause of data loss is accidental deletion of a file by the end user on their own PC, or by IT professionals of a great many files on a server. To recover from these inevitable cases of *shooting yourself in the foot*, make a backup just before loading your gun.

A **backup** is a **copy** in a **geographically separate location** on an **independent platform**. A good backup location is Microsoft Office 365 OneDrive, in a folder that is *not* synchronized with any other system. Another is to collect your data into a zip archive, and SFTP it to the matrix server.

a. Create a backup folder/directory on the target system.
b. Copy important files to that folder. e.g. the zip archive you created in Part 2. Because it is already compressed into a single file, it will take a minimum amount of time to upload.
c. Congratulations. You just backed up something.

➔ paste a screen shot of your backup results. (use the Screen Snip tool) **(10 points)**

Imagine your laptop just stopped working and could not be restarted
after you completed a great many hours of work today and yesterday.
You need a backup & restore strategy.
(30 points total for four answers ~100 words each, 400 in total.)

➔ What is (or what should have been) your backup routine? How do you ensure your backup is current?

➔ How does your backup routine address the three characteristics of a real backup and fulfill the 3-2-1 backup check?

➔ Now that you have a backup *but no laptop*, how will you access and work with the current version of your backed up files? What is your restore/recovery strategy?

➔ How long would this all take…and what if you a had a big assignment due tomorrow?

FYI – https://www.google.com/search?q=7zip+full+and+differential+backup+script