Part A: Compression (40 points)

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 algorithm used by LZW or Huffman encoding)

old quote from Vangie Beal, managing editor of Webopedia…
Only lower-case letters are used to simplify this 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.

!data *@compression* *#communications* *$transmit* *% there are* **technique* *&standard*	Dictionary Characters (with spaces) = 75
*!@is particularly useful in #because it enables devices to $ or store the same amount of !in fewer bits.%a variety of !@&s, but only a few have been *ized. the ccitt has defined a * !@& for $ting and a @* for !#through modems. in addition,%file @formats, such as arc and zip.*	Compressed text Characters (with spaces) = 275

- For the ***length*** of any string occurring ***n*** times and replaced by a single character token…
- Saving ***length*** × ***n*** occurrences has the cost of a dictionary entry (token + ***length***) plus ***n*** tokens replacing the string in the original text.
- `"data "` has a length of 5 with 5 occurrences (25 characters)
- less the overhead of 1+5 characters in the dictionary: `"!data "` plus 5 `!` tokens in the text replacing each `"data "` string
- ***length*** × ***n*** – 1 – ***length*** – ***n*** = ***x*** characters saved
- 25 saved less 6 for dictionary entry less 5 for text-to-token replacements = 14 characters saved.

Token	String	Length	Occurrences	Saving $length \times (n - 2) - 1$

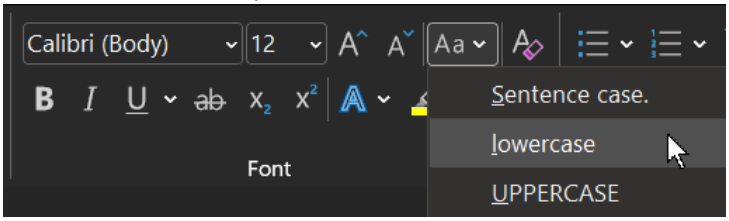
!	data	5	5	14
@	compression	12	5	42
#	communications	15	2	12
$	transmit	8	2	5
%	there are	11	2	8
&	standard	8	3	12
*	technique	9	2	6
	Original text	449	Total Saved	99
	dictionary and compressed text	350		
	compression	78%		22%

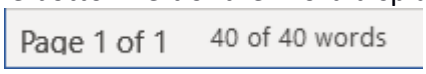
Including dictionary, total compressed size is 346 characters which is 77% of original, 23% saved.
The compression factor increases according to the frequency of repeating strings in the text.
Although Huffman and LZW compression routines are far more sophisticated, the above illustrates the concept.

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

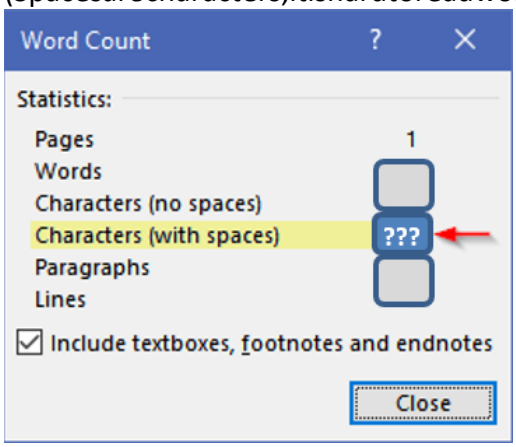
You choose the song.

→ 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. 

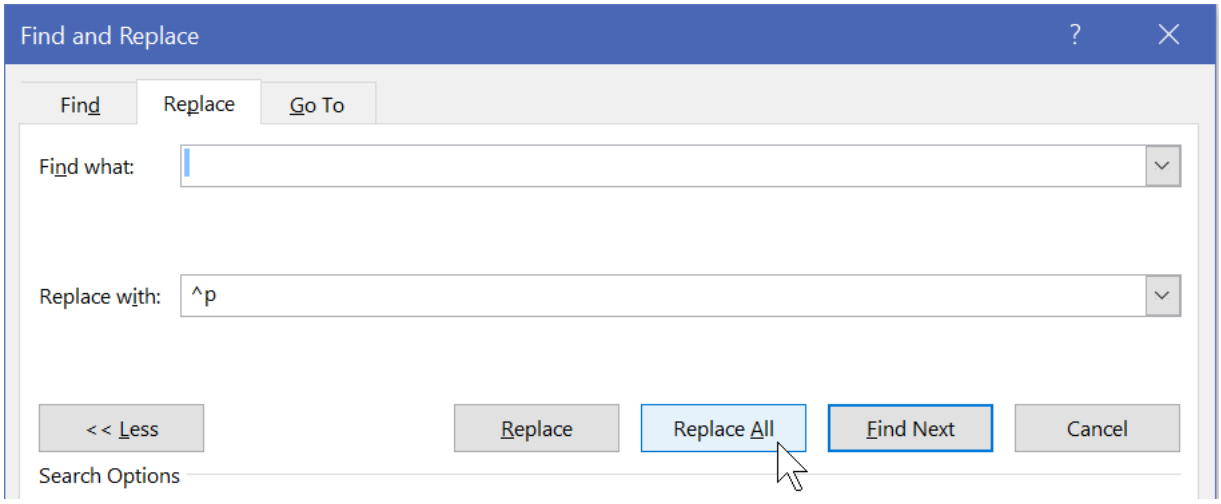
The Word Count dialog will pop up showing the number of characters with spaces.
(Spaces*are*characters,itishardtoreadwordswithoutthem.)



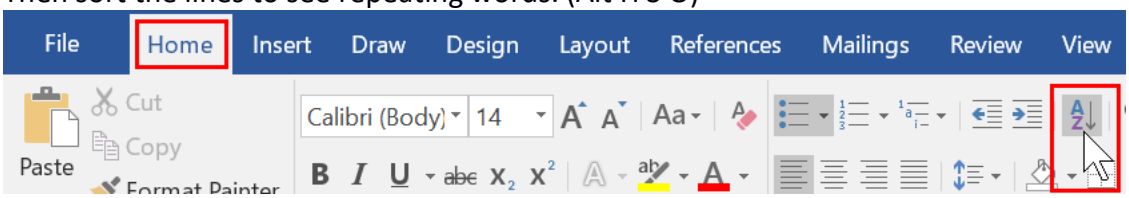
N.B. paragraph / new line / CRLF / formatting codes are not counted by Word which is fine. Our exercise here is concerned only with the text. (Alt+H,8 will toggle the display of whitespace characters)

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.

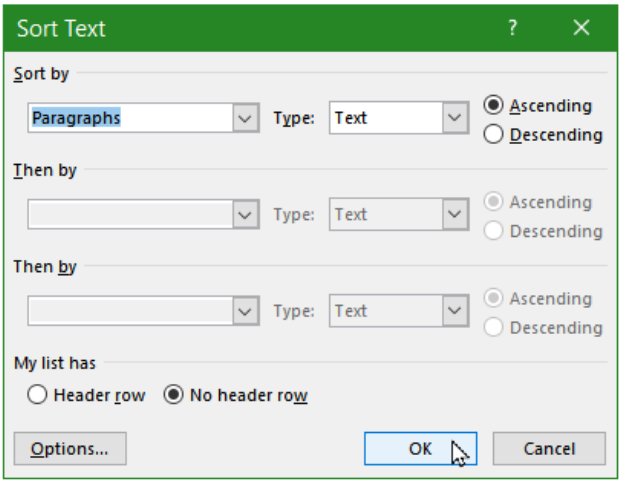
- copy the lyrics to another new document (Ctrl-N) used only for analysis
- Find and Replace a **space** with a **space + paragraph marker ^p** (Ctrl+H)
  Find what: □ Replace with: □^p



- 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 most word strings
- a robotic replacement of recurring words will not result in the best compression.
    - Consider compressing phrases before compressing individual words
    - Consider whether a leading and/or trailing space should be compressed with a word

    - Consider using portions of a word which may result in more substitutions. E.g. "transmit" and "technique" in the above example.
- the token/string dictionary must be included to decompress text back to its original state.
    - The overhead of the dictionary must be considered with the compressed text to assess the size of the compression versus the original.

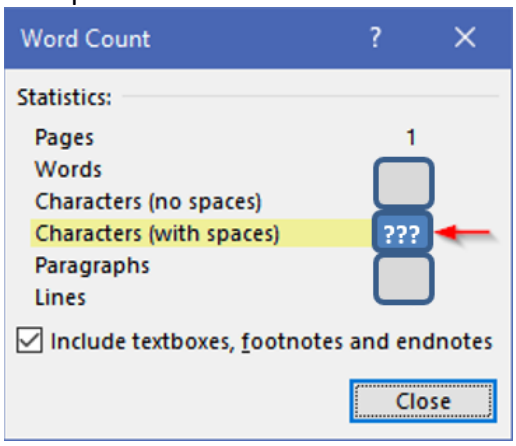
→ **How much can you compress the lyrics?**

- 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 a dictionary line as the token and the next characters to end-of-line as the original string to replace tokens with.

➔ 1. Paste the lyrics of your chosen song – with attribution please – into _Activity_Answers.docx

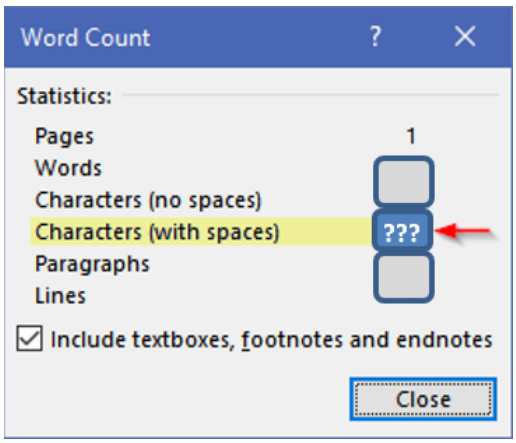
Use the CP4P_Compression_Activity_calculator.xlsx Excel file (in the archive) to help with your optimisation calculations.

→ Update the Excel file with "Characters (with space)" count **before** compression.



→ **After** your compression efforts, update Excel with the compressed "Characters (with spaces)" count. Ensure you include both the dictionary and compressed text (include trailing blank) in the count. Check that the Excel calculation of total characters saved agrees with the sum of characters saved from the

detail calcs.



➔ 2. After your compression efforts, select columns A – F (dictionary rows & totals & calculated cells) from the CP4P_Compression_Activity_calculator Excel spreadsheet and paste in _Activity_Answers.docx

➔ 3. Select the dictionary and compressed text and paste into the table in _Activity_Answers.docx

➔ 4. **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*.

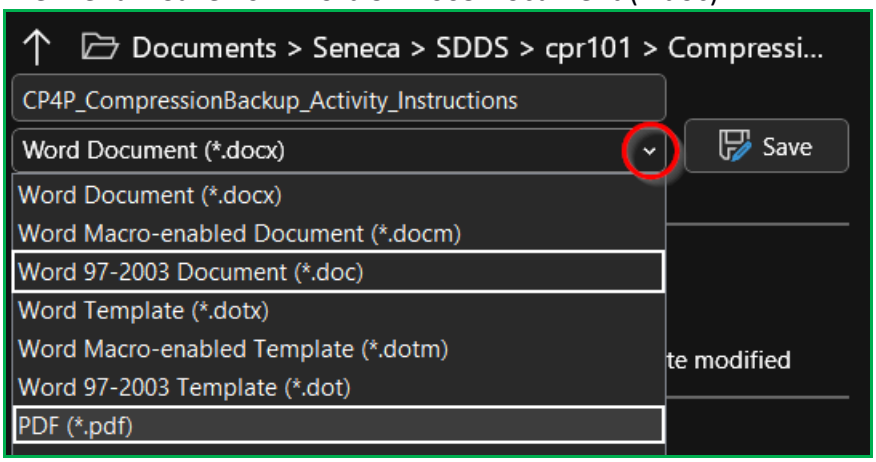
➔ 5. **What modifications, if any, does the compression dictionary need to return the compressed data back into its original state? (If none, then well done! but please make a note.)**

Part B: File formats with built-in compression…or not. (20 points)

- Download this week's activity .zip archive to your folder.
  - Remember that compressed files must be decompressed before they can be opened.
  - Windows does this automatically into the %temp% folder if you open a file directly from a .zip archive. This is fine to quickly browse a file's content.
  - However, if the file is to be kept or its content modified, first copy/extract it from the .zip archive to your folder.
    - That will be the case for `CP4P_CompressionBackup_Activity_Answers.docx` because you will be adding your answers to it. *First* extract it from the archive to your folder, *then* open it. If you open it first – into %temp% – you may never find your work again.

These next steps will open files, then save them as slightly different types. We'll do that to examine the size differences of different file format types when we add them to the archive.

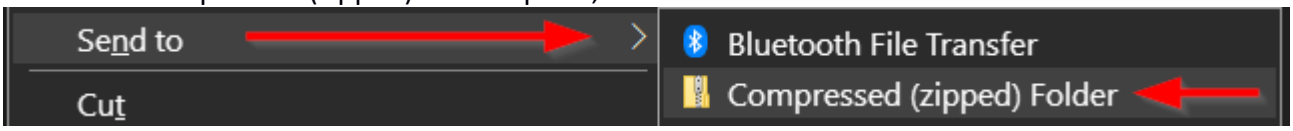
- Open the `CP4P_CompressionBackup_Activity_Instructions.docx` file.
  - File menu > Save As > Word 97-2003 Document (*.doc)



If you see the Microsoft Compatibility Checker, click Continue.

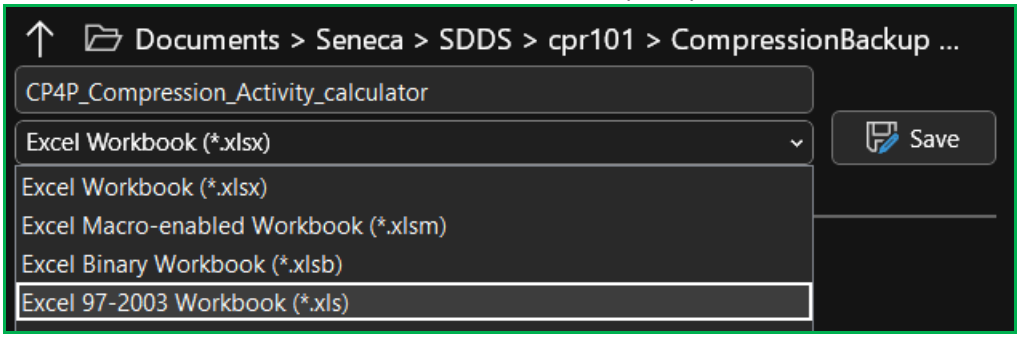
  - File menu > Save As > PDF (*.pdf)

- Add the `CP4P_CompressionBackup_Activity_Instructions.pdf` and `.doc` files from your folder into the zip archive we have been using.
  - select the files, right click, and
    Send to > Compressed (zipped) folder option,



    or use 7zip [on Windows], or your favourite compression utility to drag and drop.

- Open the `CP4P_Compression_Activity_calculator.xlsx` file.
  - File menu > Save As > Excel 97-2003 Workbook (*.xls)



If you see the Microsoft Compatibility Checker, click Continue.

  - File menu > Save As > PDF (*.pdf)

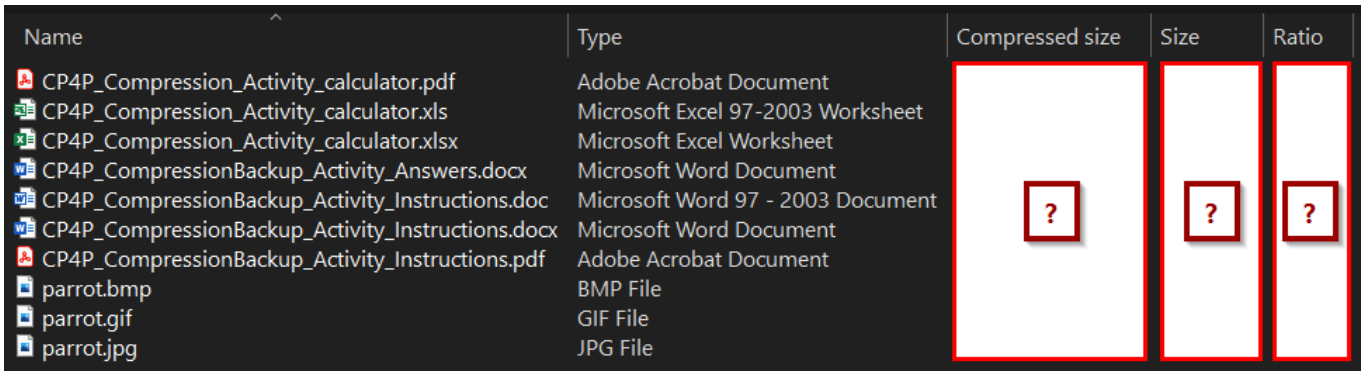
- Add the `CP4P_Compression_Activity_calculator.pdf` and `.xls` files from your folder into the zip archive we have been using.

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 % 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. That column indicates % of space saved by compression.

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

➔ **6.** Paste the image of the Windows [File] Explorer .zip archive information or equivalent from macOS into _Activity_Answers.docx.

→ 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

➔ 7. 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)

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

Part C: Backup (40 points)

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 good backup software option for ICT people is Duplicati. The downsides should not bother SDDS students. Senecans do not need a storage provider because we already have one: MS 365 OneDrive. See this review.

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, save it to a local USB drive, 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.

➔ 9. paste screen shots showing the locations and contents of your backups into _Activity_Answers.docx. (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.)

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

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

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

➔ 13. Estimate how long this would take…and what if you a had a big assignment due tomorrow?