**Name: Nicholas Defranco**

**Student id: 106732183**

**Section: CPR101SPP**

**Information about dictionary**

|  |  |  |  |
| --- | --- | --- | --- |
| (has trailing whitespace) | # of tokens in compressed text | Sum of tokens in comp. text and | Chars saved from |
| entry | Also # of occurrences of word in orig. | length of word in dictionary | original |
|  |  |  |  |
| @and  (5) <- chars | **3** | **8** | 4 |
| 3bitsy  (7) | **2** | **9** | 3 |
| 4came  (6) | **2** | **8** | 2 |
| 8itsy  (6) | **2** | **8** | 2 |
| )out  (5) | **2** | **7** | 1 |
| 9rain  (6) | **2** | **8** | 2 |
| \_spider  (8) | **3** | **11** | 10 |
| +spout  (7) | **2** | **9** | 3 |
| !the  (5) | **8** | **13** | 19 |
| $up  (4) | **3** | **7** | 2 |
|  |  |  |  |

🡺 your dictionary of token🡪string characters, one entry per line.

**\*\*\*one whitespace char after every entry\*\*\***

**@and**

**3bitsy**

**4came**

**8itsy**

**)out**

**9rain**

**\_spider**

**+spout**

**!the**

**$up**

🡺 rhyme with the token substitutions, i.e. the compressed text

**!83\_crawled $!water +**

**down 4!9@washed !\_)**

**)4!sun @dried $all !9**

**@!83\_went $!+again**

🡺 how many characters are in the dictionary + compressed text and   
what is it as percentage of the original’s 187?

**There are 139 characters including the dictionary (token and string characters with one trailing whitespace) and compressed text.**

**(139/187) \* 100% ≈ 74.33%**

**139 characters is approximately 74.33% of the original 187 characters**

🡺 Now test your compression algorithm by decompressing and pasting the decompressed version below. All algorithms must be tested. How did that work out?

**Compressed Text**

**@and**

**3bitsy**

**4came**

**8itsy**

**)out**

**9rain**

**\_spider**

**+spout**

**!the**

**$up**

**!83\_crawled $!water +**

**down 4!9@washed !\_)**

**)4!sun @dried $all !9**

**@!83\_went $!+again**

1. **Replace all occurrences of ‘!’ with “the ”**

**the 83\_crawled $the water +**

**down 4the 9@washed the \_)**

**)4the sun @dried $all the 9**

**@the 83\_went $the +again**

1. **Replace all occurrences of ‘8’ with “itsy ”**

**the itsy 3\_crawled $the water +**

**down 4the 9@washed the \_)**

**)4the sun @dried $all the 9**

**@the itsy 3\_went $the +again**

1. **Replace all occurrences of ‘3’ with “bitsy ”**

**the itsy bitsy \_crawled $the water +**

**down 4the 9@washed the \_)**

**)4the sun @dried $all the 9**

**@the itsy bitsy \_went $the +again**

1. **Replace all occurrences of ‘\_’ with “spider ”**

**the itsy bitsy spider crawled $the water +**

**down 4the 9@washed the spider )**

**)4the sun @dried $all the 9**

**@the itsy bitsy spider went $the +again**

1. **Replace all occurrences of ‘$’ with “up ”**

**the itsy bitsy spider crawled up the water +**

**down 4the 9@washed the spider )**

**)4the sun @dried up all the 9**

**@the itsy bitsy spider went up the +again**

1. **Replace all occurrences of ‘+’ with “spout ”**

**the itsy bitsy spider crawled up the water spout**

**down 4the 9@washed the spider )**

**)4the sun @dried up all the 9**

**@the itsy bitsy spider went up the spout again**

1. **Replace all occurrences of ‘4’ with “came ”**

**the itsy bitsy spider crawled up the water spout**

**down came the 9@washed the spider )**

**)came the sun @dried up all the 9**

**@the itsy bitsy spider went up the spout again**

1. **Replace all occurrences of ‘9’ with “rain ”**

**the itsy bitsy spider crawled up the water spout**

**down came the rain@washed the spider )**

**)came the sun @dried up all the rain**

**@the itsy bitsy spider went up the spout again**

1. **Replace all occurrences of ‘@’ with “and ”**

**the itsy bitsy spider crawled up the water spout**

**down came the rain and washed the spider )**

**)came the sun and dried up all the rain**

**and the itsy bitsy spider went up the spout again**

1. **Replace all occurrences of ‘)’ with “out ”**

**the itsy bitsy spider crawled up the water spout**

**down came the rain and washed the spider out**

**out came the sun and dried up all the rain**

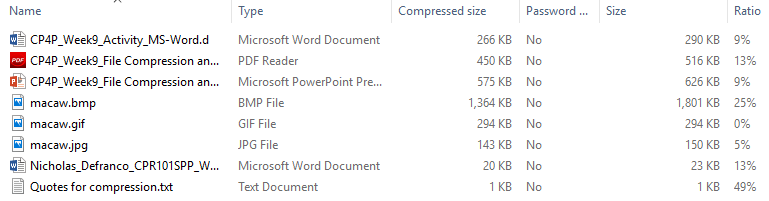
**and the itsy bitsy spider went up the spout again**

**Part 2:**

Paste the image of the Windows Explorer .zip archive information here…

Use the Snipping Tool ( + “snip”) to copy only the information seen above.

**🡺**



What do the compression ratio percentages tell you about the contents of the different kinds of files? Were some of the files already compressed in their original format before zipping tried to compress them further? Which files do you think were already compressed? **(10 points)  
🡺**

If the ratio percentage is large, it probably means that the original file size was large as well since there is a lot more content that could be compressed (more possible occurrences of the same pattern which can be replaced by a symbol – run length encoding (Chapman 2018)). Such as the macaw stored as a bitmap on disk as well as the text file, both of which are large files with a large ratio.

The photo of the macaw stored as a gif on disk has a ratio of 0% meaning that it did not compress at all when I compressed and put them into a zip archive. This is possibly indicating that it just could not compress as it may have been to small for the file format compression algorithm to reduce.

* Used source you have provided.

The files with high Ratios were compressed the most. Which files were they? Why were they compressed the most? **(10 points)**See <https://www.noupe.com/design/everything-you-need-to-know-about-image-compression.html>   
🡺

The macaw stored as a bitmap on disk as well as the text file “Quotes for compression.txt” both have a significantly higher ratio percentage than the other files in the zip archive.

The bitmap compressed significantly because for bitmaps, there are a lot of repeated patterns which is why the process run-length encoding is used where every pattern that the algorithm sees in the file that repeats more than once is assigned a specific symbol along with a counter determining the number of occurrences (Chapman 2018). Since patterns are repeated a lot this process saves a lot of space.

* Used source you have provided.

The text file has compressed the most out of all the files because there is a method called the Huffman coding which sorts every character in a hierarchical structure by their rarity. Common characters are assigned a smaller binary number while rarer ones are assigned a larger binary number (Thompson 2017). This has been proven to be the best way save storage with text files as it stops the need to represent a character always with one byte (ASCII char) to instead only using exactly the number of bits required to represent the character which the tree creates.

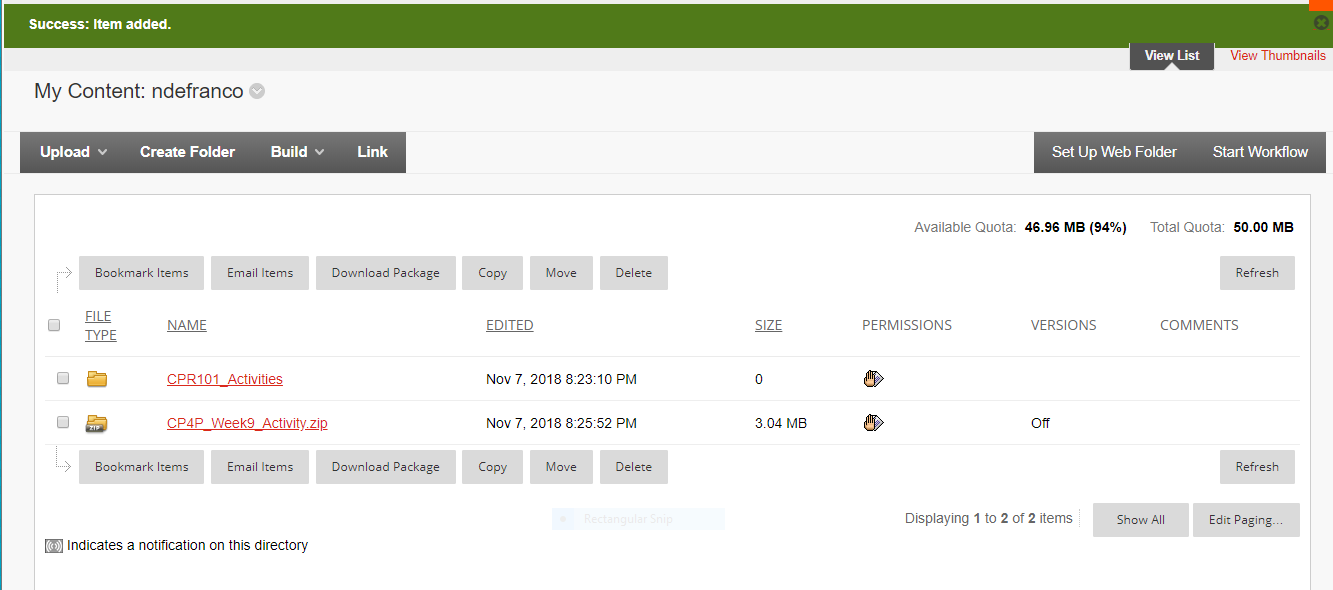
Source for more information on Huffman coding:

<https://www.popularmechanics.com/technology/a28154/how-do-we-compress-text-files/>

Had a good video too

**Part 3: Backup**

🡺 paste a screen shot of your backup results. (Alt-PrtScn or use the Snipping 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. What is (or what should have been) your backup & restore strategy? How does your strategy address the 3 characteristics of a real backup? How does your strategy fulfill the 3-2-1 backup check? Replacing a machine is more than just restoring your lost data files. What about the Operating System and the applications which process those files? How long would this all take…and what if you a had a big assignment due tomorrow? **(30 points)**🡺

For an assignment that has a series of questions such as this one, my backup and restore strategy should be for every question answered (more info and how and where it is saved in 3 characteristics paragraph).

For a longer assignment such as a c program, I will save the project often and not keep a lot of data in the computer’s buffer and risk losing it (write it disk). After every day I do a backup to a secondary drive (USB). If it happens to be the most stable version, I will increment the minor (or major if the change is significant) count as well as state the description of the version of the file in comments in the code. The micro versions of the code will be kept as backup until the minor version count has been incremented as by then, it will hopefully be properly tested so there is no longer a need for them (just for safety micro versions will be kept even when a major version has been made unless the project is far in development and I am sure they are not needed). Depending on the changes it will average about 2 weeks which then I will perform a full backup as described in the 3 characteristics paragraph.

3 characteristics paragraph

I will save it as new version to my local hard drive as well as USB Flash drive and to the “cloud” preferably google drive or matrix if it is code (since I can easily perform a secure copy to other locations or perform secure file transfer protocol). Each of which will be separate of each other and that if one fails, I still have the other copies that did not depend on the first that went missing (deleted) or corrupted. Which is saved in a completely unrelated location from the original that was copied. Every time I create a new version of the file, it will increment the version number (depending on the change). Each version will be copied to my backup locations when it is time to backup to be as safe as possible.

3-2-1 backup check

3 copies – local hard drive (C drive), USB, google drive or matrix if programming (using scp to transfer files to there or from there).

2 different formats/platforms – USB can back up files regardless of the platform

* Google drive copies files when you drag and drop files.

1 off-site backup – I can save files to another computer by transferring through a USB, copy from USB to other computer.

If you had to replace your machine and had to install Microsoft applications if you want to continue working locally. Instead of waiting to download the applications off the Microsoft store (as well valid your account and so on) which could take a while depending on the internet connection, google drive is much easier to access as you can visit it from any browser. It supports Microsoft word (docx) format and you can continue editing on google docs (SaaS). For editing word documents google docs can be just as capable as Microsoft word. For programming, instead of trying to reinstall visual studio, I can log in to my matrix account and use Vi IMproved to continue programming. It is a powerful text editor that has a lot of features making it easy to edit file. It also colour codes source code very nicely (better than visual studio colour coding in my opinion) making it easy to read code. It’s also much faster to make quick changes to a file in VIM than it is in visual studio.