SARANSH AGARWAL (2019MT60763)

HARSH SHARMA (2019MT60628)

ELL786 REPORT

ASSIGNMENT 2

**Experiment 1:**

**Lempel–Ziv–Welch** (**LZW**) is a universal [lossless data compression](https://en.wikipedia.org/wiki/Lossless_data_compression) [algorithm](https://en.wikipedia.org/wiki/Algorithm) created by [Abraham Lempel](https://en.wikipedia.org/wiki/Abraham_Lempel), [Jacob Ziv](https://en.wikipedia.org/wiki/Jacob_Ziv), and [Terry Welch](https://en.wikipedia.org/wiki/Terry_Welch). It was published by Welch in 1984 as an improved implementation of the [LZ78](https://en.wikipedia.org/wiki/LZ77_and_LZ78) algorithm published by Lempel and Ziv in 1978.

Implement the LZW encoder

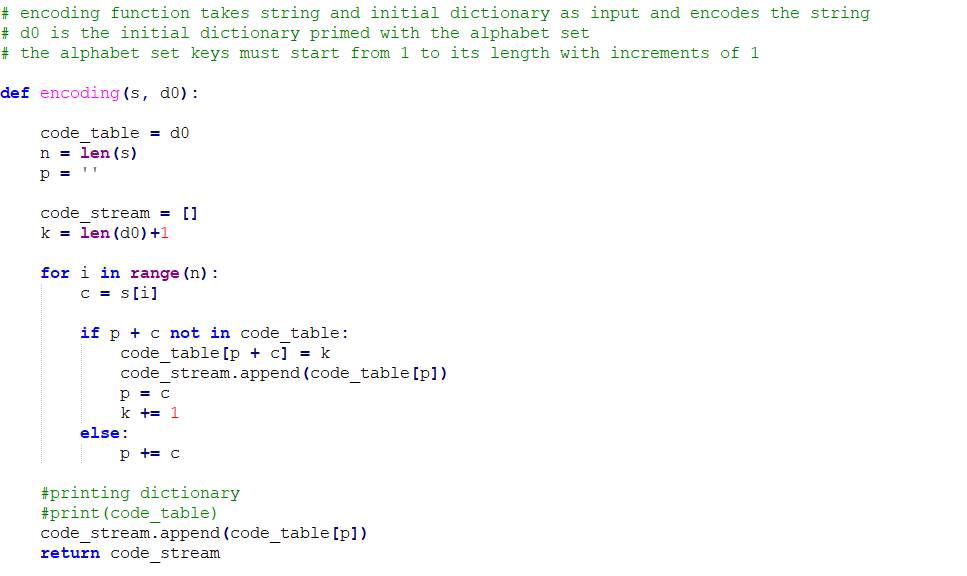


Figure 1: Wrapper Code

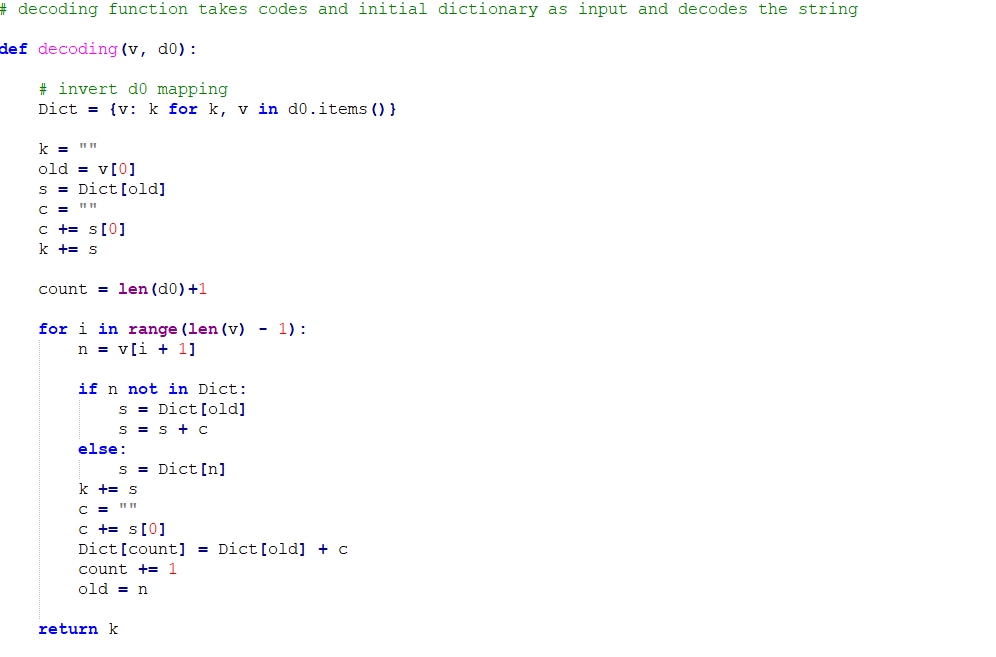


Figure 2: Wrapper Code

Testing against an initial dictionary consisting of the letters a, b, r, y, ·, then encode the

**“a· bar· array· by· barrayar· bay”** message using the LZW algorithm..

|  |  |
| --- | --- |
| INDEX | ENTRY |
| 1 | a |
| 2 | b |
| 3 | r |
| 4 | y |
| 5 | · |

Figure 3: Initial Dictionary

|  |  |
| --- | --- |
| INDEX | ENTRY |
| 1 | a |
| 2 | b |
| 3 | r |
| 4 | y |
| 5 | · |
| 6 | a· |
| 7 | ·b |
| 8 | ba |
| 9 | ar |
| 10 | r· |
| 11 | ·a |
| 12 | arr |
| 13 | ra |
| 14 | ay |
| 15 | y· |
| 16 | ·by |
| 17 | y·b |
| 18 | bar |
| 19 | rr |
| 20 | ray |
| 21 | ya |
| 22 | ar· |
| 23 | ·ba |

Figure 4: LZW dictionary for encoding message

The encoded sequence for the message **“a· bar· array· by· barrayar· bay”** is

1,5,2,1,3,5,9,3,1,4,7,15,8,3,13,4,9,7,14

**ENCODING BMP FILE**



Figure 5: bmp file to be encoded

Now, convert the bmp file to a binary string

Text, letter

Description automatically generated

Figure 6: Convert the bmp file to a binary string

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 7: Wrapper Code

**CONCLUSION:**

Length of binary string before compression = 39321600 bits

Length of binary string after compression = 13359700 bits

**Compression ratio = 39321600/13359700 = 2.94**

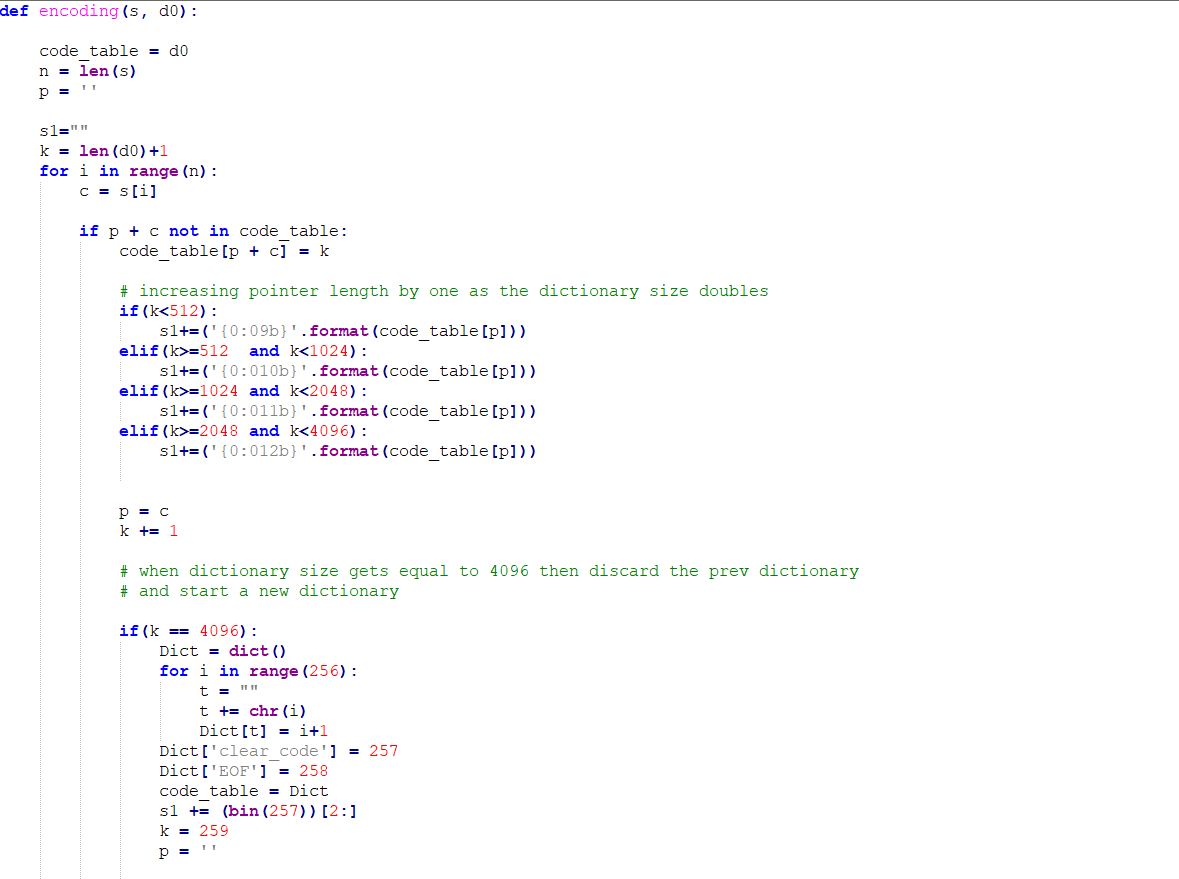
|  |  |  |  |
| --- | --- | --- | --- |
| **Encoding scheme** | **Length of binary string before compression(bits)** | **Length of binary string after compression(bits)** | **Compression Ratio** |
| LZW | 39321600 | 13359700 | 2.94 |
|  |  |  |  |

Figure 8: compression ratio for LZW

**Experiment 2:**

The **Graphics Interchange Format** is a The Graphics Interchange Format is a bitmap image format that was developed by a team at the online services provider CompuServe led by American computer scientist Steve Wilhite and released on 15 June 1987. It has since come into widespread usage on the World Wide Web due to its wide support and portability between applications and operating systems. that was developed by a team at the online services provider CompuServe led by American computer scientist Steve Wilhite and released on 15 June 1987. It has since come into widespread usage on the World Wide Web due to its wide support and portability between applications and operating systems.

Implement the GIF encoder



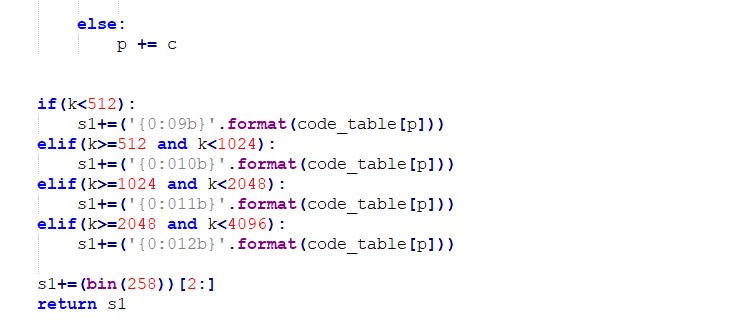


Figure 9: Wrapper Code

**ENCODING BMP FILE**



Figure 10: bmp file to be encoded

Now, convert the bmp file to a binary string

Text, letter

Description automatically generated

Figure 11: convert the bmp file to a binary string

Text

Description automatically generated

Figure 12: Wrapper Code

**COMPRESSION RATIO:**

Length of binary string before compression = 39321600 bits

Length of binary string after compression = 20321360 bits

**Compression ratio = 39321600/20321360 = 1.93**

**COMPARISION BETWEEN LZW AND GIF ENCODER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoding scheme** | **Length of binary string before compression(bits)** | **Length of binary string after compression(bits)** | **Compression Ratio** |
| LZW | 39321600 | 13359700 | 2.94 |
| GIF | 39321600 | 20321360 | 1.93 |
|  |  |  |  |

Figure 13: compression ratio for LZW and GIF

**CONCLUSION:**

From figure 13 we can conclude that compression ratio by LZW technique is 0.66 while the compression ratio by GIF encoder is 0.48. The compression ratio for GIF is less than LZW. In GIF we use two additional special codes clear codes and End of Information codes which is not there in LZW. In GIF the pointer gets incremented by one from dictionary to dictionary due to which GIF has variable compression sizes. In GIF the dictionary becomes static as the size of dictionary becomes equal to 4096. Due to which in GIF the computational complexity increases. That's why we are getting less compression ratio for GIF than LZW.

**Additional techniques used in gif:**

The Graphics Interchange Format (GIF) was developed by CompuServe Information Service to encode graphical images. It is another implementation of the LZW algorithm and is very similar to the compress command.GIF uses dynamic dictionary for compression.

Algorithm:

* It takes number of bits per pixel b as parameter.
* It uses initial dictionary of size 2^(b+1). When this dictionary fills up, we doubled the dictionary size until we reach maximum dictionary size of 4096.At this point dictionary behaves as static dictionary.
* The codewords from the LZW algorithm are stored in blocks of characters. Pointer get longer one byte from dictionary to dictionary and output are in block of 8 bytes.
* After reaching maximum size of dictionary 4096 we discard the old dictionary and start new dictionary.
* At the time of starting new dictionary, Encoder emits **clear code** which is sign for decoder to discard the dictionary.
* The end of compressed file is denoted by an **END OF INFORMATION** code.

**Experiment 3:**

Implement the PNG encoder

PNG is a purely lossless format initially intended to replace the GIF format. It is based on a predictor followed by the dictionary compression method DEFLATE (a combination of LZ77 and Huffman coding), which is usually implemented by the zlib library.

Text

Description automatically generated

Figure 14: Wrapper Code

**ENCODING BMP FILE**



Figure 15: bmp file to be encoded

Now, convert the bmp file to a binary string

Text, letter

Description automatically generated

Figure 16: convert the bmp file to a binary string

**COMPRESSION RATIO:**

Length of binary string before compression = 39321600 bits

Length of binary string after compression = 11501485 bits

**Compression ratio = 39321600/11501485 = 3.42**

**COMPARISION BETWEEN LZW AND PNG ENCODER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoding scheme** | **Length of binary string before compression(bits)** | **Length of binary string after compression(bits)** | **Compression Ratio** |
| LZW | 39321600 | 13359700 | 2.94 |
| PNG | 39321600 | 11501485 | 3.42 |
|  |  |  |  |

Figure 17: compression ratio for LZW and PNG

**CONCLUSION:**

From figure 17 we can conclude that compression ratio by LZW technique is 0.66 while the compression ratio by PNG encoder is 0.92. The compression ratio for PNG is more than LZW. For lossless compression, a better method is to predict pixels based on neighbouring pixels already seen, then compress the residual prediction error. PNG uses this format, using deflate to compress the residual. That's why we are getting more compression ratio for PNG than LZW.

**Additional techniques used in PNG:**

The PNG standard is one of the first standards to be collaboratively developed over the Internet. The impetus for it was an announcement in December 1994 by Unisys (which had acquired the patent for LZW from Sperry) and CompuServe that they would start charging royalties to authors of software that included support for GIF.

* The PNG encoder uses deflate implementation of LZ77 technique.
* In PNG encoder the match length is between 2 and 258.
* In PNG technique at each step three bytes of data is examined. If we can’t find a match of at least three bytes then we put first byte and examined next three bytes.
* At each step we put value of single byte or the pair <match length, offset >.
* The total alphabet size is 286 The indices 0-255 represent literal bytes and the index 256 is an end-of-block symbol. The remaining 29 indices represent codes for ranges of lengths between 3 and 258.
* The index values are represented using a Huffman code.
* The offset takes values between 1 and 32678.These values are divided into 30 ranges and these 30 range values are encoded using Huffman codes.

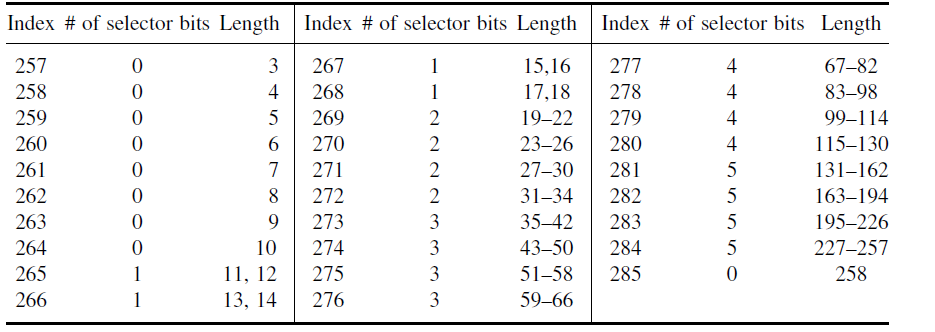


Figure 16: code representation for match length 59.

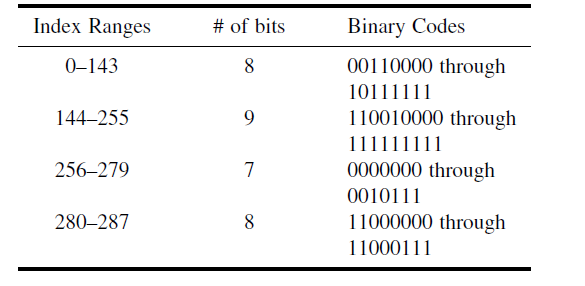


Figure 18: Huffman codes for match length 59.

**COMPARISION BETWEEN LZW, GIF AND PNG ENCODER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoding scheme** | **Length of binary string before compression(bits)** | **Length of binary string after compression(bits)** | **Compression Ratio** |
| LZW | 39321600 | 13359700 | 2.94 |
| GIF | 39321600 | 20321360 | 1.93 |
| PNG | 39321600 | 11501485 | 3.42 |
|  |  |  |  |

Figure 19: compression ratio for LZW, GIF and PNG

**CONCLUSION:**

**Experiment 4:**

**USING JPEGLS ENCODER**

JPEG-LS (1998) is a lossless image format created as a replacement for the lossless JPEG mode, which is inefficient. This compression method consists of a predictor and a subsequent context entropic coder (Golomb-Rice coder). It’s implementation is provided in python in the following code snippets and the attached zip file which contains actual source code

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Figure 20: Wrapper Code

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Figure 21: Wrapper Code

Graphical user interface, application

Description automatically generated

Figure 22: Wrapper Code

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

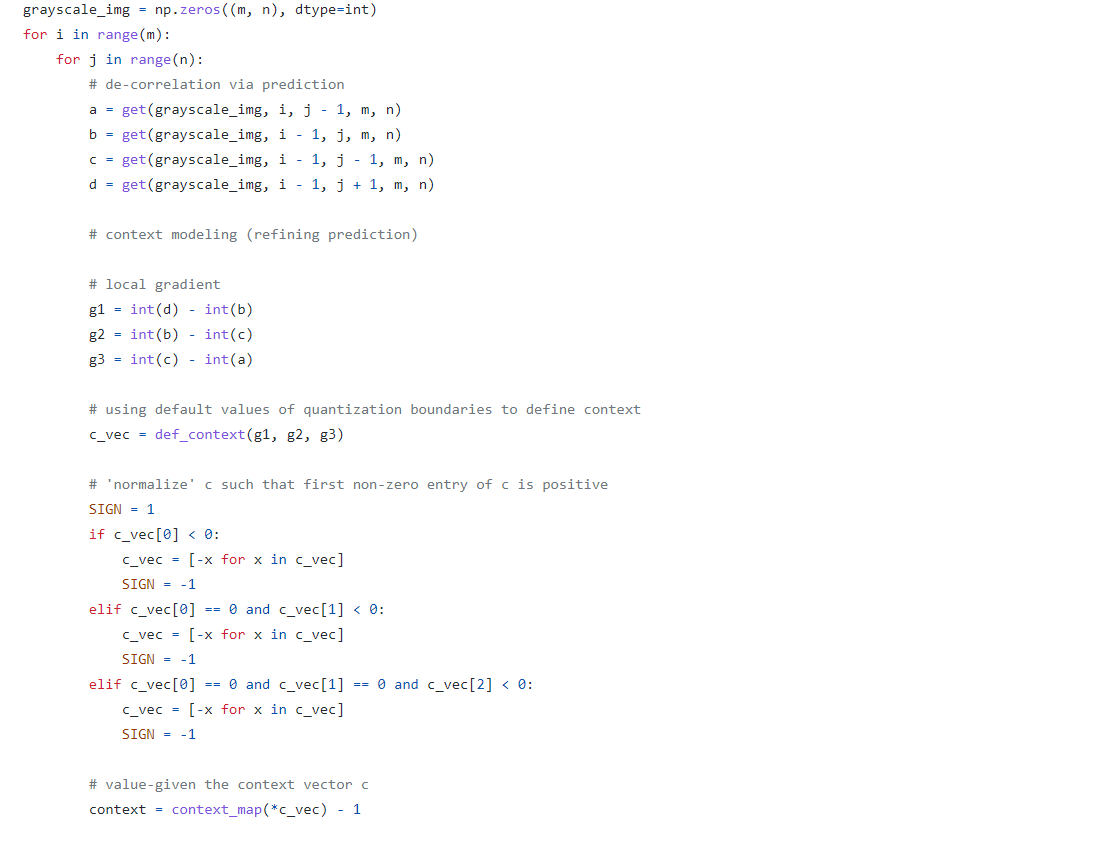
Graphical user interface, text, application

Description automatically generated

Figure 23: Wrapper Code

Graphical user interface, text, application

Description automatically generated



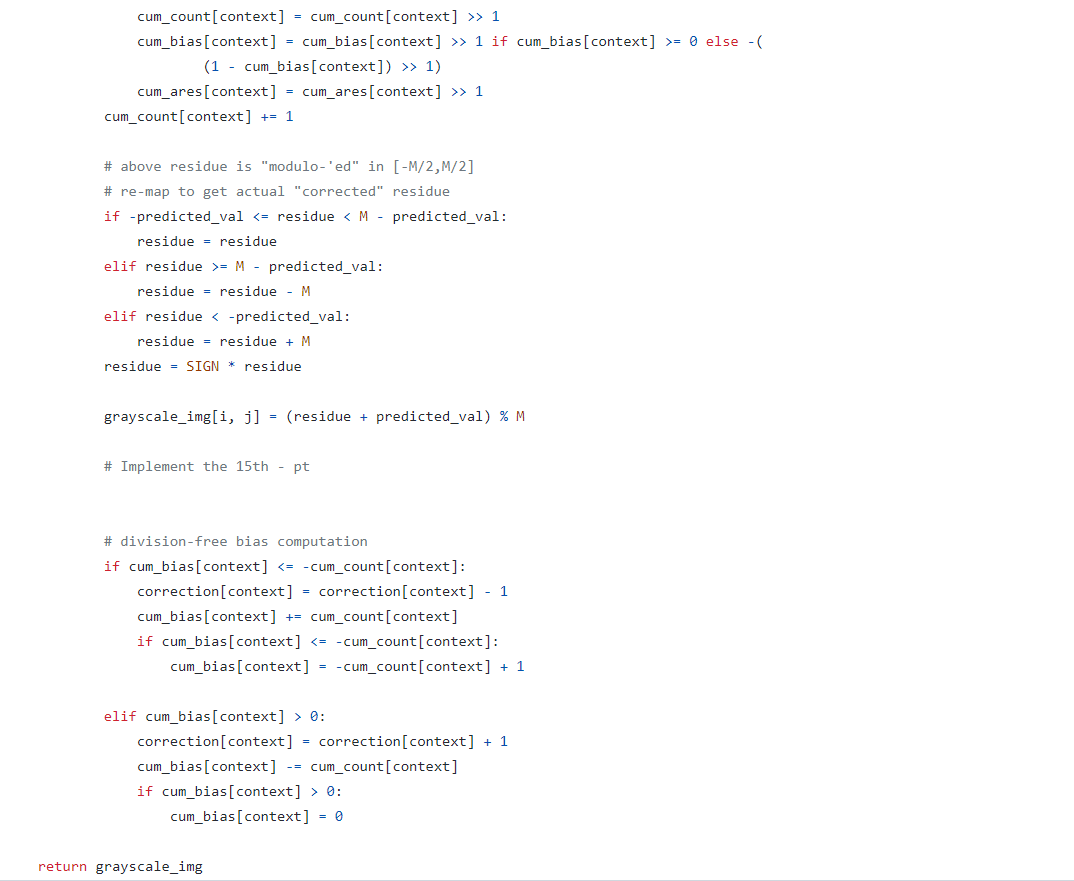


Figure 24: Wrapper Code

**Graphical user interface, text, application

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

Figure 25 : Wrapper Code

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

Figure 26 : Wrapper Code

A picture containing dark

Description automatically generated A picture containing dark

Description automatically generated A picture containing dark

Description automatically generated

Original Image JPEG-LS Compressed Image PNG Compressed Image

Figure 27 : Original vs JPEG-LS vs PNG

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

Figure 28 : Wrapper Code

**CONCLUSIONS**

1. Similar to how we compressed earth image, to compare PNG and JPEG-LS we compressed multiple images and summarize their result in the following table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image | Original Size | JPEG-LS | PNG | CR(JPEG-LS) | CR(PNG) |
| Lena | 90037 KB | 33829 KB | 69337 KB | 2.66 | 1.30 |
| Earth | 13824 KB | 5120 KB | 6137 KB | 2.70 | 2.25 |
| Sea | 22443 KB | 13788 KB | 16061 KB | 1.62 | 1.39 |
| Nature | 7763 KB | 6146 KB | 6444 KB | 1.26 | 2.20 |
| Facebook logo | 16911 KB | 3819 KB | 3527 KB | 4.43 | 4.80 |
| Coffee | 1780 KB | 871 KB | 649 KB | 2.04 | 2.74 |
| Spiderman | 3037 KB | 1252 KB | 1095 KB | 2.42 | 2.77 |

*These images (original, jpeg compressed, png compressed) can be found in the images directory inside assignment zip folder*

We can observe that, for more natural palletized images, JPEG-LS seems to do better than PNG.

For dithered, halftoned, and some graphic images for which LZ-type methods are better suited PNG does better than JPEG-LS

1. **PSNR** (Peak Signal To Noise Ratio) for JPEG-LS and PNG compressed images is not defined as these techniques are *lossless compression techniques* and hence observed **MSE** (Mean Squared Error) in both JPEG-LS and PNG is 0.0 (which is indicated in the output of Fig. 28 code snippet)