A brief analysis of the application

The application has the implementation of the rice algorithm encoder and the function to compress the audio files.

The function RiceAlgoencode() implements the rice algorithm which helps to encode the audio file.

When a sound file is passed into the encode function, a list of frames of the sound file will be passed into the function together with the k (number of bits). The modulo is found by taking 2 to the power of k. The remainder is than found by taking S modulo the modulo value. The quotient is then obtained. The quotient is looped through and 1 is added to the quotient code. After loop ends, a zero is added at the end of the quotient code. The binary format is then obtained from the remainder. Code word is found by adding the quotient code and binary format.

```
def RiceAlgoencode(S,k):
    M_odulo = 2 ** k
    r_emainder = S % M_odulo
    quo_tient =int(math.floor(S / M_odulo))
    Quo_C = ""


    for i in range(quo_tient):
        Quo_C += "1"
    Quo_C += "0"

    binary_FORMAT = "{0:b}".format(r_emainder)



    Code_WORD = Quo_C + binary_FORMAT


    return Code_WORD
```

The function compress() helps to compress the audio files obtaining the byte array of the audio file.

```
def compress_SOUNDFILE(sound_file,k):

    sound_file = wave.open(sound_file, mode="rb")
    frames_list = list(sound_file.readframes(sound_file.getnframes()))
    frames_ = bytearray(frames_list)

    f = open("Sound1_Enc_2bits.ex2", "w")
    for i in range(len(frames_)):
        f.write(RiceAlgoencode(k, frames_[i]))
        f.write('\n')
    f.close()
```

The encode function is applied to the frames of the sound file.

```
compress_SOUNDFILE("Sound1.wav", 2)
```

The sound file and the number of bits (k=2) is passed into the compress function. The compress function will read and extract the frames in the sound file. The frames will be placed in a byte array. After looping through the byte array list, each frame will be encoded together with the number of bits. And the newly encoded files will be written into a new file and in this case, it is called Sound1_Enc_2bits.ex2.

The same method of compression and encoding is applied to sound 1 when k =4, sound 2 when k =2 and k=4.

The requested table and an analysis of the results. Justify why the compression rates of the files are so different.

	Original Size	Rice (K=4 bits)	Rice (K=2 bits)	% Compression (K = 4 bits)	% Compression (K = 2 bits)
Sound1.wav	1002088 bytes	5349344 bytes	4193022 bytes	433%	310%
Sound2.wav	1008044 bytes	5057599 bytes	4048385 bytes	402%	302%

The compression rate of the sound 1 when k=2 bits is 310% and when k = 4 it is 433%. The compression rate is higher when the value K is higher as during encoding, when a higher K value is used, the binary format that is returned for the encode is bigger, therefore, it causes the encoded file to have a bigger size, thus a bigger compression rate.

The same goes for sound 2. The compression rate of sound 2 when k=2 is 302% and when k=4 it is 402%. The binary format returned by the encode function will be longer when k =4 compared to when k =2. Therefore, the file which was encoded with 4 bits had a bigger size than the file that was encoded with 2 bits, thus leading to a higher compression rate.

In conclusion, the longer the binary format, the higher the size of the file and thus resulting in a higher compression rate.

A brief description of the further development implemented.

The rice algorithm implemented here is good when the encoded audio is played back, the quality of the audio remains the same and this indicates that this is a lossless data compression, that is during the conversion process, there is no loss of information. One thing that can be improved on is the size of the file after compression. Using this compression method and the encoding algorithm resulted in very big files. That's the thing that can be improved upon**.** Unfortunately, I couldn't implement this