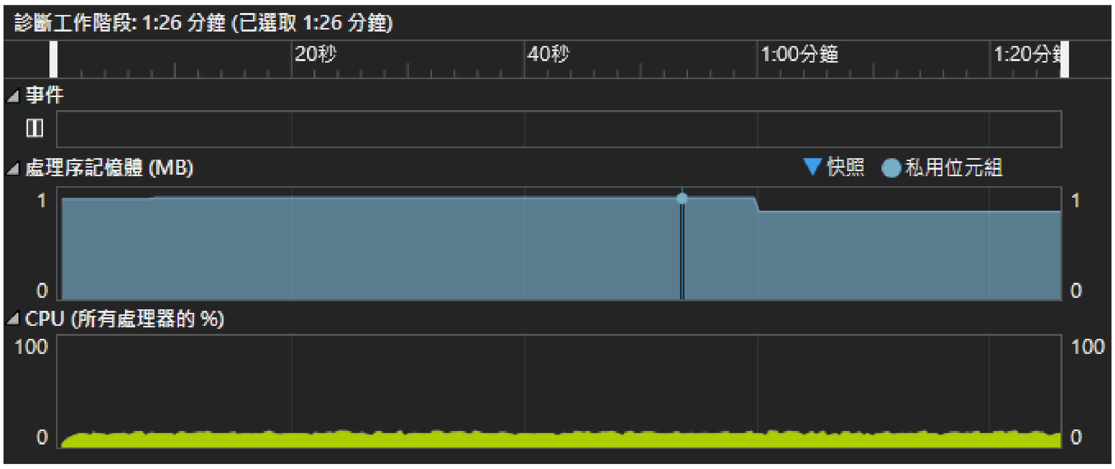
ID Number: 11363111 Name: 陳瑞鑫

Program File	**Image_Huffman.cpp**
Encode Input File	**lenna.bmp:** 512 * 512, gray, bmp file.
Encode Output File	**bmpHeader.txt:** Binary bmp file header + palette. **huffTable.txt:** 0 – 255 Huffman code. Store by string. **lennaCompression.txt:** Binary encode bmp array.
Decode Input File	**bmpHeader.txt:** Binary bmp file header + palette. **huffTable.txt:** 0 – 255 Huffman code. Store by string. **lennaCompression.txt:** Binary encode bmp array.
Decode Output File	**lenna_r.bmp**: Decode from huffTable.txt, lennaCompression.txt. Header equal lenna.bmp.
Time	**Debug mode:** About 90 Seconds (contain decode and encode). **Release mode**: About 5 Seconds (contain decode and encode).
Memory (Debug mode)	About 1 MB.
CPU	Intel Core I7-9700K
RAM	16 GB

Debug mode:

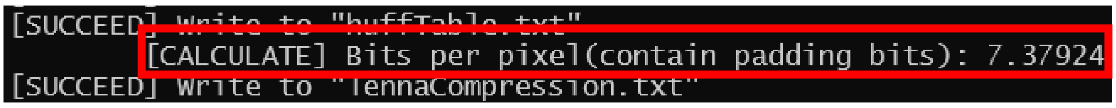


Q1: Use C/C++ implement a Huffman coder(decoder)

Please reference Image_Huffman.cpp.

Q2: Use Encoder to encode lenna.bmp, and calculate bits per pixel

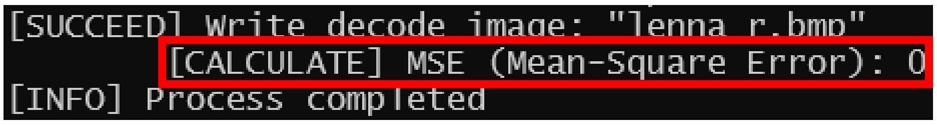
Please reference output



```
[SUCCEED] Write to "huffTable.txt"
        [CALCULATE] Bits per pixel(contain padding bits): 7.37924
[SUCCEED] Write to "lennaCompression.txt"
```

Q3: Decode to lenna_r.bmp, and calculate lenna.bmp and lenna_r.bmp MSE (Mean-Square error)

Please reference output



```
[SUCCEED] Write decode image: "lenna_r.bmp"
        [CALCULATE] MSE (Mean-Square Error): 0
[INFO] Process completed
```

Q4: YouTube URL:

https://www.youtube.com/watch?v=qH4l8KlRuRM

Timeline(**YouTube 影片說明欄** / Back up content from video descriptions):

0:00 展示程式執行過程 / Demo

0:10 計算題 / bits per pixel and MSE (Mean-Square Error)

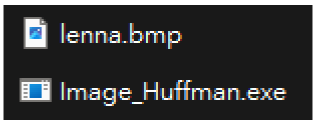
0:25 比較 lenna.bmp 跟 lenna_r.bmp / Show lenna.bmp and lenna_r.bmp HEX

0:40 Show bmpHeader.txt, huffTable.txt, lennaCompression.txt

1:14 Show lenna.bmp and lenna_r.bmp

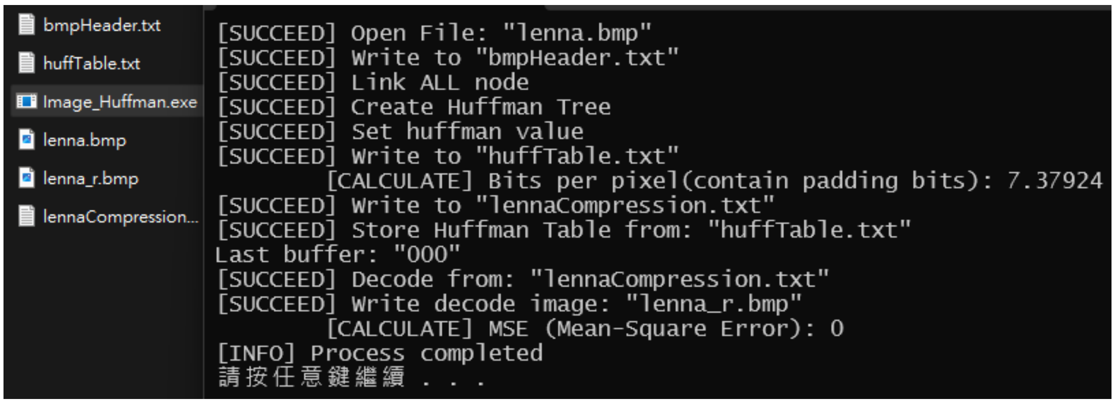
✧ Execute:

- cd "Image_Huffman/**Release**"
- Make sure contain "lenna.bmp" and "Image_Huffman.exe"



```
lenna.bmp
Image_Huffman.exe
```

- Execute Image_Huffman.exe



```
bmpHeader.txt
huffTable.txt
Image_Huffman.exe
lenna.bmp
lenna_r.bmp
lennaCompression...
```

```
[SUCCEED] Open File: "lenna.bmp"
[SUCCEED] Write to "bmpHeader.txt"
[SUCCEED] Link ALL node
[SUCCEED] Create Huffman Tree
[SUCCEED] Set huffman value
[SUCCEED] Write to "huffTable.txt"
        [CALCULATE] Bits per pixel(contain padding bits): 7.37924
[SUCCEED] Write to "lennaCompression.txt"
[SUCCEED] Store Huffman Table from: "huffTable.txt"
Last buffer: "000"
[SUCCEED] Decode from: "lennaCompression.txt"
[SUCCEED] Write decode image: "lenna_r.bmp"
        [CALCULATE] MSE (Mean-Square Error): 0
[INFO] Process completed
請按任意鍵繼續 . . .
```